University of Zurich[UZH]

Department of Informatics

StakeCloud: Stakeholder Requirements Communication in the Cloud

Dissertation submitted to the Faculty of Business,
Economics and Informatics
of the University of Zurich

to obtain the degree of
Doktorin der Wissenschaften, Dr. sc.
(corresponds to Doctor of Science, PhD)

presented by
Irina Koitz
from Romania

approved in April 2016

at the request of
Prof. Dr. Martin Glinz
Prof. Dr. John Mylopoulos

University of Zurich[UZH]

The Faculty of Business, Economics and Informatics of the University of Zurich hereby authorizes the printing of this dissertation, without indicating an opinion of the views expressed in the work.

Zurich, April 6, 2016

Chairwoman of the Doctoral Board: Prof. Dr. Elaine M. Huang

Acknowledgments

To see far is one thing, going there is another.

Constantin Brâncuși

I have always had the courage and enthusiasm to see far, but the resources to go there have not always been exclusively mine. I am profoundly grateful to all the amazing people who have encouraged and guided me along my path.

First, I would like to express my deepest gratitude to my supervisor, Prof. Martin Glinz, for trusting and giving me the freedom to explore a completely new research area. I appreciate that he always supported me in pursuing my research ideas, including those that even I considered wild. His detailed advice and guidance always brought back the light when the going got tough. I am very happy to have had an advisor with whom I synchronized so well and who greatly contributed to my becoming the person I am now.

Special thanks also go to my external supervisor, Prof. John Mylopoulos, for accepting to be part of my doctoral committee

and for his early feedback on my work as part of the Doctoral Symposium Committee at RE'12.

I would also like to acknowledge the Swiss National Science Foundation (SNSF), which partially supported my PhD work (grant no. 200021_150142). Furthermore, my appreciation goes to all the collaborating cloud provider companies and students who contributed to the StakeCloud project. Thank you, Martina, Mihaylo, Matthias, Louis and Osman for your great work!

One of the reasons why I am here now is that some passionate academics inspired me to do research during my Master studies at Aalto University (Helsinki University of Technology). Kiitos paljon, Marjo, Heikki, Mika, Matti and Kari!

To my RERG colleagues from whom I learned so much about research and beyond, with whom I shared ideas but also conferences and fun times, thank you for everything! Big thanks, Cédric, Eya, Norbert, Reinhard and Anne for guiding me as a fresh doctoral student and teaching me what a PhD is about, big thanks, Dustin, Martina, Parisa, Sofija, Emitzá, Rita, Neda and Melanie for the great discussions about papers, cultures, life and the world, and for being there for me during these years. I am also grateful to all my other friends and colleagues from the Department of Informatics and Institute of Computational Linguistics at University of Zurich who always made Binzmühlestrasse 14 an awesome workplace to come to every day. Thank you, Barbara, for joining me in my mission to support women in computing, for motivating me to (temporarily) become a screenwriter and film actor and for your tireless and positive spirit!

To my friends from near and far, from Romania to Finland and from the Netherlands to Japan, thank you so much for always listening to me, still liking me even when I am difficult and being by my side notwithstanding the distance.

I would have never come to this point in my life without the continuous and unconditional love and support from my family. I wholeheartedly thank my parents for the way they raised me to appreciate the clean and the beautiful and for everything they invested in me. Last but definitely not least, I thank my husband Ralph for encouraging me to do a PhD, for reviewing my papers and my thoughts and for constantly believing in me.

To you all, vă mulțumesc din suflet, vielen Dank, merci vielmal, many thanks!

Irina
Zürich, January 2016

Abstract

Effective requirements communication between consumers and providers represents the foundation of any successful service or product. Therefore, numerous methods have been developed and used throughout the recent decades to support provider companies in knowing their consumers and understanding their needs.

However, the existing requirements communication techniques are seriously challenged by the cloud paradigm. Cloud computing has emerged as a successful service delivery model that allows reducing capital costs, while improving service accessibility, flexibility and scalability. As a result, researchers and practitioners focused on enhancing the technological capabilities of cloud services, but the topic of gathering consumers' real needs has largely been ignored. Challenges posed by the cloud model such as wide, heterogeneous and geographically distributed audiences, frequent change requests and short times to market can only marginally be supported by conventional requirements communication methods.

In this thesis, we first describe the current state of practice with regard to requirements communication in cloud settings. The

results of an exploratory study we conducted with 19 cloud service providers show that most companies use ad-hoc methods for gathering consumer requirements, since the existing techniques do not fit the cloud characteristics. Furthermore, we identified what key features a cloud requirements communication method should have to meet providers' needs.

Secondly, we present StakeCloud, a novel dedicated cloud requirements communication approach. StakeCloud is the main contribution of the thesis and has three components: a conceptual solution, its practical implementation, and a final evaluation. The conceptual solution has its roots in Galois theory and consists of building fuzzy Galois lattices based on (potential) consumers' advanced search queries collected online on marketplaces. The Galois lattices can be used by cloud providers to analyze market needs and trends, as well as optimum solutions for satisfying the largest populations possible with a minimum set of implemented requirements. Moreover, as proof of concept, we implemented a practical tool that can be used directly by cloud provider representatives, e.g., product managers. Finally, we evaluated to what extent our approach satisfies the main needs identified in the exploratory study. The StakeCloud approach complements the existing plethora of requirements communication techniques in that it is a dedicated method for cloud settings, operates with data that already exist, and enables large-scale consumers' involvement in an unobtrusive fashion.

Zusammenfassung

Effektive Anforderungskommunikation zwischen Kunden und Dienstleistern ist die Grundlage für erfolgreiche Dienste (Services) und Produkte. Aus diesem Grund wurden in den letzten Jahrzehnten zahlreiche Methoden entwickelt und angewandt, um Anbieter beim Verstehen von Kundenbedürfnissen zu unterstützen.

Die bestehenden Techniken der Anforderungskommunikation werden vom Cloud Computing Paradigma jedoch ernsthaft herausgefordert. Cloud Computing hat sich zu einem erfolgreichen Modell zur Servicebereitstellung entwickelt, das Kapitalkosten reduziert und Erreichbarkeit, Flexibilität und Skalierbarkeit erhöht. Folglich haben sich Forschung und Anwendung darauf konzentriert, die technologischen Fähigkeiten von Cloud-Diensten zu verbessern, während die Erhebung von realen Kundenanforderungen grossteils vernachlässigt wurde. Die Herausforderungen bei der Bereitstellung von Cloud-Diensten wie breites, heterogenes und geografisch verteiltes Publikum, häufige Änderungswünsche und kurze Vorlaufzeiten können durch konventionelle Methoden der Anforderungskommunikation nur ungenügend gemeistert werden.

In dieser Arbeit beschreiben wir zunächst den aktuellen Stand der Praxis betreffend Anforderungskommunikation im Cloud-Umfeld.

Eine Überblicksstudie, die wir mit 19 Cloud Dienstleistern durchgeführt haben, zeigt, dass die meisten Unternehmen ad-hoc Methoden zur Anforderungsermittlung verwenden, da die verfügbaren Techniken für die Cloud nicht geeignet sind. Ausserdem haben wir Merkmale identifiziert, die eine Methode zur Cloud Anforderungskommunikation aufweisen sollte, um Dienstleistern zu nützen.

Anschliessend präsentieren wir StakeCloud, einen dezidierten Ansatz zur Anforderungskommunikation für die Cloud. StakeCloud ist der Hauptbeitrag dieser Arbeit und hat drei Komponenten: ein Lösungskonzept, dessen praktische Implementierung und eine abschliessende Evaluation. Das Lösungskonzept hat seine Wurzeln in der Galoistheorie. Aus Suchanfragen von Kunden auf online-Marktplätzen konstruieren wir unscharfe Galoisverbände (fuzzy Galois lattices). Diese können von Cloud-Anbietern genutzt werden, um Marktbedürfnisse und Trends zu analysieren und Lösungen zu finden, welche eine grösstmögliche Zahl von Kunden mit einem minimalen Satz an implementierten Anforderungen zufriedenstellen. Darüber hinaus haben wir als Nachweis der praktischen Machbarkeit ein Werkzeug implementiert, welches Mitarbeiter von Cloud-Anbietern, z.B. Produktmanager, direkt nutzen können. Abschliessend evaluieren wir, in welchem Ausmass unsere Lösung die in unserer Überblicksstudie aufgezeigten Hauptbedürfnisse erfüllt. StakeCloud ergänzt die bestehende Vielfalt von Techniken zur Anforderungskommunikation, da es spezifisch für Cloudumgebungen ist, mit existierenden Daten arbeitet, und in einfacher Weise umfangreiche Kundenbeteiligung ermöglicht.

Contents

List of Tables

List of Figures

Chapter 1
Synopsis

1.1 Introduction

Effective requirements communication between consumers and providers [GW07] is essential for building successful services and products [SS97]. Ignoring or incorrectly addressing end users' or consumers' needs generally yields solutions that do not solve the problems they were designed for, thus leading to increased costs [Poh10] and failure-prone projects [Tuu03, AW05]. However, consumers' requirements are not always easy to identify: some are difficult to express, some are hidden, while some may overlap or conflict. To address these challenges, software and requirements engineering (RE) have provided excellent support and methods to elicit requirements [ZC05] throughout the recent decades. In general, requirements elicitation refers to seeking, gathering and consolidating requirements and is the first step in the requirements engineering process [NE00].

Despite the successful application of existing elicitation techniques so far, numerous assumptions need to be rethought today. The advent of *cloud computing* [AFG+09] and its novel technical and business model for delivering software and hardware resources flexibly, on demand [BYV+09], in an elastic fashion [MG11], has brought opportunities unknown before. In this context, cloud services are solutions that are not owned by clients any more, as in typical traditional settings, but are delivered by cloud providers.

Glinz and Wieringa defined stakeholders as persons or organizations that have a direct or indirect influence on the requirements of a system [Gli15, GW07]. In cloud settings, the main stakeholders are the cloud service consumers and providers. According to the US National Institute of Standards and Technology [LTM+, p. 7], a *cloud (service) provider* is an organization that "acquires and manages the computing infrastructure required for providing the services, runs the cloud software that provides the services, and makes arrangements to deliver the cloud services to the cloud consumers through network access". Conversely, a *cloud (service) consumer* is "a person or organization that maintains a business relationship with, and uses the service from a cloud provider" [LTM+, p. 5].

Embracing the emerging cloud computing paradigm, providers can sell their solutions worldwide to large masses of consumers, at a scale that was unimaginable in traditional settings [BYV+09], where both providers and consumers were collocated or at least within easy reach. Nevertheless, this also brings new challenges from a requirements communication perspective: cloud consumers

are mostly physically unreachable, globally distributed and highly heterogeneous [Vou08]. They often have frequent change requests and their requirements are volatile [Vou08], since early cloud adopters are modern dynamic companies or individuals [ZRB15]. Moreover, providers cannot rely on knowing local markets since they often sell their services in locations where they do not have and cannot easily send their consultants or requirements experts. Finally, the cloud is still young, which means that service providers cannot build on existing expertise as far as cloud consumers' requirements communication or elicitation is concerned [Vou08]. Therefore, the existing requirements elicitation methods are slowly rendered obsolete because core prerequisites these techniques build upon (e.g., stakeholders are known and reachable) are no longer true in a cloud computing context. At most, traditional approaches can only partially support the requirements elicitation activity in isolated cases [RZWT12, TSG13]. Hence there is a lack of methods that fit cloud providers' needs with regard to identifying and understanding their (potential) consumers' requirements. Researchers and practitioners have recognized the importance of the cloud, but they have mostly focused on improving the technical aspects of cloud offerings so far, virtually ignoring the actual end-users, the cloud consumers [IH10, KAD10].

Yet satisfying consumers' needs is not possible without knowing what their actual needs are. Further, delivering solutions that match cloud consumers' requirements is too important of a business goal to remain without concrete support and appropriate solutions. Although much effort has been dedicated to developing requirements elicitation techniques and ways to choose the

most suitable ones in particular situations, the current state of the art cannot properly support cloud service providers in understanding their consumers. This represents a serious *stakeholder requirements communication problem in cloud settings*. There is a general mismatch between the cloud solutions offered and the actual consumer needs, which frequently leads to dissatisfaction for both main stakeholders involved: service providers do not manage to sell their solutions successfully and consumers do not receive services that match their needs.

1.2 State of the Art

The widely recognized importance of stakeholder requirements communication has resulted in a considerable number of requirements elicitation techniques, approaches and tools. For instance, Zowghi and Coulin provided a comprehensive survey of existing work applicable in traditional systems [ZC05]. Furthermore, many researchers also observed the need to choose specific techniques depending on individual settings, and described strengths and limitations of elicitation approaches in different contexts [HD03, LRA02, Tuu03].

In this section, we review the state of the art in requirements elicitation techniques that have potential to be applied in cloud settings. Firstly, we present the current views and approaches that specifically target the cloud domain with regard to communicating stakeholders' requirements. Then, as requirements elicitation in

distributed and market-driven settings is related to elicitation in cloud computing contexts, we subsequently survey the existing body of work in these areas. Finally, we explain how the current knowledge gap in requirements communication approaches for cloud contexts motivates our work, show where a new dedicated cloud method fits in the existing plethora of requirements elicitation techniques, and conclude with our thesis statement.

1.2.1 Requirements Elicitation in Cloud Computing Settings and Related Challenges

As the cloud paradigm is still rather young, requirements engineering for the cloud is currently lagging behind [SW11]. However, both researchers and practitioners have recognized the importance of understanding cloud consumers' requirements. For instance, Koehler et al. remarked that good services can only be delivered when consumers' needs are taken into account and that exclusively adopting a technical perspective on improving the cloud offering will not lead to successful solutions [KAD10, KADW10]. Moreover, Vouk emphasized the need for appropriate methods to elicit consumer requirements, since this is the only way to generate value in the cloud [Vou08]. Nevertheless, achieving this is not a simple task [CPK10]. Chen et al. identified serious problems which can arise due to misunderstanding the roles, needs and interests of cloud stakeholders. According to them [CPK10], relationships in the cloud can be more complicated than the traditional provider-user relationship, which makes it vital to ensure good cloud requirements communication. Therefore, it is commonly agreed that

there is a lack of dedicated requirements elicitation approaches which can be used effectively by cloud providers to understand their consumers' needs [ZB11].

As a result, numerous researchers aimed at solving the cloud consumers' requirements communication problem in various ways. However, most of these attempts are rather limited. For instance, some tried to strictly satisfy consumers from a service functionality perspective, completely disregarding nonfunctional requirements. Nevertheless, gathering and analyzing nonfunctional requirements is essential, since it has long been acknowledged that omitting or misinterpreting them commonly leads to the most expensive and difficult to correct errors [MCN92]. By nonfunctional requirement, we understand an attribute of or a constraint on a system [Gli07], where the attributes can be performance requirements (e.g., speed, throughput) or specific quality requirements (e.g., security, availability). The approaches in the direction of eliciting nonfunctional requirements are few, typically look at exclusively one type of nonfunctional requirement and rather pose a challenge than provide a solution. For example, Iankoulova and Daneva conducted a systematic review of cloud computing security requirements [ID12], Gritzalis and Liu looked into requirements engineering for security and privacy [GL13], Mouratidis et al. developed a framework supporting the selection of cloud providers based on security and privacy requirements [MIKG13] and Kalloniatis et al. evaluated cloud deployment scenarios based on security and privacy requirements [KMI13]. Notwithstanding, none of these approaches brings concrete solutions for cloud providers to understand their con-

sumers' needs and restrict themselves to only presenting research directions.

Furthermore, other approaches are limited in that they only consider one type of cloud service, e.g., SaaS (Software as a Service) [BELK10], completely ignoring other types of cloud computing such as IaaS (Infrastructure as a Service) or PaaS (Platform as a Service). As a result, the existing cloud offers do not meet or only partially meet consumers' requirements, thus leading to dissatisfaction and low revenue, as observed by Repschlaeger et al. [RZWT12]. Therefore, the question that still remains unanswered is how to cover multiple dimensions, including both service functionality and nonfunctional requirements, to gain a complete understanding of cloud consumers' needs.

The main reason why this is still an open and difficult issue lies in the native characteristics of the cloud. These make it virtually impossible for cloud service providers to use existing requirements elicitation methods or the experience accumulated over the years when building successful conventional systems.

The *number of stakeholders* in cloud settings is often beyond what traditional methods can support. On the one hand, the existing approaches cannot easily handle *large masses of consumers* [Vou08], and novel, flexible and scalable techniques are required [ZB11]. On the other hand, *stakeholders' involvement* is a recurring problem in requirements engineering which should not be neglected in the cloud context either. Numerous researchers regard user involvement to be critical for the success of a system [WB13, NK09, KKLK05]. Since most existing elicitation

approaches require the direct and conscious participation of stakeholders, providing incentives for users to get involved generally becomes necessary. Cloud service consumers are most often located remotely, outside reasonable physical reach, thus motivating them to participate in requirements elicitation activities becomes even more problematic [Kuj03]. Lichtenstein et al. [LNH07] proposed retrieving requirements from service level agreement (SLA) documents as a means for enabling consumer involvement. This is done from the perspective of end-users who can consult SLAs and decide whether the specified services meet their needs or not. However, this approach can only work for service consumers that are known individually and who are able to correctly define their requirements. Therefore, this solution would only support a small percentage of cloud service consumers. Another problem of this solution is the potential lack of SLAs. There are situations when no SLA documents are provided, but rather general descriptions of cloud service performance. Thus, an SLA-based approach would not work in such a case.

Trienekens et al. [TBvdZ04] proposed to directly enable communication between consumers and providers by using a phased process for specifying SLAs. Although this work contributes to the field of requirements communication, it has a significant limitation: requirements are communicated only when negotiating SLAs, i.e. after a consumer has selected a provider. Thus, this approach does not support early requirements communication with potential consumers. Moreover, similar to the work of Lichtenstein et al. [LNH07], this approach also relies on the existence of SLA documents.

Furthermore, the cloud computing setting is characterized by *globally distributed consumers*. Thus, traditional approaches (such as interviews or workshops) which require synchronous interaction with representative stakeholders are difficult or, in some cases, even impossible to apply in this context [LRA02, Saw00].

Moreover, unlike the traditional settings where stakeholders are usually individually identifiable and within organizational reach [Saw00], the cloud is characterized by *heterogeneous potential consumers* who are often *unidentifiable* on an individual basis.

Additionally, conventional software-intensive systems undergo a managed evolution. Naturally, stakeholders' requirements evolve, triggering the evolution of entire systems [SLAM13]. However, in the cloud, there are no strict borders and no real evolution constraints: user needs can continuously change [Her07]. Whereas cloud services are designed to easily scale from a technical perspective, providers need to become aware of the requirements changes before being able to fulfill them. This is not a trivial task when employing only the existing requirements elicitation methods and the difficulty of responding to change requests and *unstable and volatile consumer requirements* are therefore often seen as a key challenge [Vou08]. Additionally, cloud providers have to deliver services *fast* to meet the changing market requirements. For this, researchers observed the need to have structured methods that support requirements management and looked into the dynamic aspect of requirements in the cloud [ZRB15]. However, they only focused on cloud adoption and no attention has been dedicated to early RE activities such as elicitation.

Other researchers investigated what development process models are suitable for cloud contexts, such as the V model or Extreme Programming (XP), and proposed a comparison framework that includes these, evaluating how they fit the cloud characteristics [SW11]. Schrödl and Wind mentioned that the requirements elicitation phase, as a first step, is crucial to understand what stakeholders want. However, they concluded that providers should use well-known methods such as interviews, workshops and scenarios, without discussing their suitability in cloud settings. Similarly, Ramachandran analyzed how to gather cloud service requirements related to enterprise-wide business objectives [Ram13], and he also suggested conducting interviews, focus groups and ethnographic studies. Nevertheless, he did not evaluate these methods in a cloud context, and these approaches are widely seen as virtually impossible to use due to the cloud characteristics, as already noted.

We can conclude that there have been attempts to use and adapt the existing requirements elicitation techniques to cloud contexts. Yet no conclusive results have been obtained so far and most researchers argued that new techniques that fit the challenges posed by the cloud paradigm are needed. In this regard, only a few conceptual models, vague ideas and numerous research directions constitute the state of the art for dedicated cloud requirements elicitation methods in 2015.

The situation is equally poor when it comes to selecting cloud providers. Cloud consumers usually choose their service providers using ad-hoc approaches, based on recommendations from friends or based on the reputation of provider brands [ZBE14]. As a

result, marketplaces [MTG11, BK14, Int15, AT05, Clo15] emerged as community platforms that allow (potential) cloud consumers to input their needs and, based on these, matching cloud solutions from various providers are suggested. The needs are generally input as advanced search queries. An *advanced search query* is a query that a (potential) cloud consumer enters into a marketplace in a structured form, i.e. (s)he specifies desired values for a given finite set of service features.

Cloud community platforms are still at their very beginning now, but it is forecast that they will increasingly grow and gain importance in the near future [FR15]. However, despite the large amounts of consumer search queries collected on such platforms, these data have never been used to enable the communication of requirements from cloud consumers to providers, such that service suppliers can then offer solutions well-tailored to the real needs of the market.

1.2.2 Requirements Elicitation in Distributed Settings

Research in distributed requirements engineering shows that due to its communication- and collaboration-intensive nature, requirements elicitation becomes particularly difficult compared to traditional settings [DZ03]. As cloud systems are distributed by their very nature, research results for requirements elicitation in distributed settings naturally extend to cloud computing.

For instance, researchers looked into the possibility to use and/or adapt well-known elicitation techniques to distributed settings, and evaluated their success. For example, Lloyd et al. [LRA02] conducted a study on the effectiveness of elicitation techniques in distributed requirements engineering. This study included well-known, traditional techniques such as interviews, questionnaires and brainstorming. The conclusion drawn was that a synchronous elicitation approach is generally more effective than an asynchronous one. However, setting up a synchronous approach in distributed and cloud settings is challenging due to the lack of collocation, thus generating the need for providing efficient asynchronous requirements elicitation methods.

Lim et al. [LQF10] presented ideas on asynchronous and distributed stakeholder identification. This approach assumes that key stakeholders are known, and others can be identified based on existing stakeholders' domain knowledge. This may lead to identifying more stakeholders of particular groups. However, identifying heterogeneous users who are not necessarily connected to the existing ones is also considered critical. Therefore, the challenge of identifying stakeholders in environments where they are highly heterogeneous should not be neglected [NE00].

Researchers also tried to tackle the problem of reaching and involving wide audience end users or users who are not within organizational reach. Tuunanen [Tuu03] argued that traditional techniques do not provide adequate solutions and presented methods which could potentially fill this gap (e.g., EasyWinWin). Nevertheless, so far, none of these methods has been successfully used on a

large scale for enabling and supporting distributed elicitation. Research on EasyWinWin by Kukreja and Boehm [KB12] promised to provide support for distributed settings, but only focused on stakeholders within organizational reach. Further, it is widely known that group elicitation techniques, such as JAD, focus groups and brainstorming, facilitate the collaboration and involvement of stakeholders in the elicitation process [ZC05]. Therefore, there have been attempts to also adapt these techniques for remote, online use [FdS09, FdS11]. Another trial in this direction is represented by the use of wikis as collaborative tools which support stakeholder involvement [DRR+07, KBKF09, SA10, YWK+08]. However, all these approaches assume that consumers are known and identifiable, which is a recognized challenge in the cloud context.

Studies from the field of web-information systems [YT03] revealed that the needs regarding Internet-based systems are also rather different from those of traditional systems. Yang and Tang pointed out that this is also caused by the users' diversity. Unfortunately, such particularities were not sufficiently considered when the current well-known methods were designed. Therefore, it is difficult to elicit requirements from consumers who have highly diverse backgrounds and needs [YT03].

In summary, most of the existing distributed requirements elicitation methods rely on the existence of stakeholders within easy reach and consider them identifiable. Therefore, these characteristics make the existing distributed elicitation techniques only marginally applicable in a cloud context.

1.2.3 Requirements Elicitation in Market-Driven Settings

According to Sawyer [Saw00], the main differences between market-driven and traditional RE settings consist of time constraints and stakeholder characteristics. These observations are also fairly relevant in the cloud context, where stakeholders and dynamics differ from those of traditional software or hardware development.

Firstly, time to market is usually an important factor which determines the choice of requirements elicitation methods to be used in a market-driven project. Development cycles are usually rather short and there is a constant flow of requirements [CB95, Hon95, KDNoD+03, APC06, DKP+03]. Therefore, elicitation techniques which require significant time and effort in order to achieve meaningful results do not fit. These constraints also apply in a cloud computing setting.

Secondly, in market-driven projects, stakeholders are usually too numerous to allow service or product suppliers to select individual persons as representatives of stakeholder roles and elicit articulated requirements from them [CB95, Hon95, KDNoD+03]. Consequently, requirements are often invented by developers or other stakeholders belonging to the developing organization [Pot95], relying on their own professional experience.

As an alternative, some companies perform market studies to understand the trends of the market and what people need or desire, and use these to develop their own products or services. In many cases,

this leads to copying what competitors do, where providers try to use already proven success recipes. Other organizations conduct in-depth studies, test their prototypes with selected stakeholders, and then generalize the results to mass markets.

In the market-driven domain, most products have some local market even when they are sold globally, such that some of the stakeholders are within reach and available for direct interaction. In these cases, traditional requirements elicitation methods can be applied, but in a limited fashion, only to part of the audience. Subsequently, based on the results, user patterns can be defined and then generalized to extended wide audiences of stakeholders.

As soon as a supplier has products or services in the market that are actually being used, feedback from users can be exploited for extracting new requirements. For instance, Seyff et al. investigated spontaneous end-user requirements blogging [SGM10a]. Additionally, bug reports and change requests can be analyzed for extracting requirements manually or using mining tools [HEGM13].

Despite being partially relevant for the cloud, the findings from the market-driven RE field are not generally applicable in a cloud computing context. This is due to the fact that cloud services by their very nature address a large number of diverse and globally distributed consumers and typically do not have a local, individually reachable customer base. Furthermore, relying on the experience of cloud providers' employees to invent requirements for new services is often seen as an unsuitable idea due to the young age of the cloud model, thus lack of extensive understanding.

1.2.4 The Need for a Dedicated Cloud Requirements Elicitation Approach: Motivation and Thesis Statement

According to the current state of the art, it is evident that the existing cloud-specific approaches are in a fairly preliminary stage and researchers most often draw research directions instead of providing concrete solutions for the cloud stakeholder requirements communication problem. Moreover, the elicitation techniques used so far in distributed and market-driven contexts are only partially applicable in cloud settings. Therefore, there is a need for dedicated cloud elicitation techniques that support cloud companies in understanding their consumers and providing them with services that meet their needs. Such methods should satisfy the challenges introduced by the cloud paradigm we presented in Section 1.2.1. Moreover, there is a need for studies that investigate how cloud service providers concretely perform the requirements elicitation activity today, in order to know what strategy to adopt and what types of methods would fit their needs.

In their work, Tsumaki and Tamai [TT06] classified the existing requirements elicitation methods according to two criteria. Firstly, depending on how requirements acquisition is conducted, requirements can be collected and sorted either in a static or dynamic way. Secondly, depending on the properties of the target space analyzed, the space can be either closed or open. Using this categorization, due to the fast and dynamic pace of the cloud, service providers should elicit requirements in a *dynamic*, ideally continuous fashion.

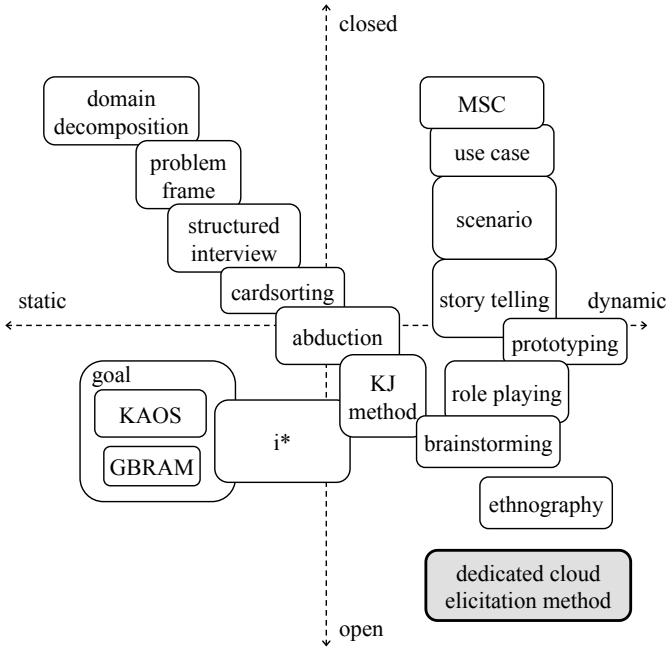

Figure 1.1: Requirements elicitation techniques, adapted after [TT06]. First published in [TKG15a].

Since consumers' needs may change rapidly and this can often be unpredictable, the space is *open*. As shown in Figure 1.1, methods such as brainstorming, role playing or ethnography could seemingly fit these characteristics. However, these are the type of methods that necessarily require the physical, synchronous and simultaneous presence of stakeholders in the same geographical space, which is incompatible with the cloud paradigm. Thus a dedicated cloud elicitation method would belong in the framework proposed

by Tsumaki and Tamai in the bottom right corner, supporting a dynamic elicitation process in an open space, as depicted in Figure 1.1.

Our core idea is that analyzing data generated by consumers while searching for cloud services and deriving new knowledge therefrom is a valuable approach for answering the problem of eliciting and understanding cloud consumers' requirements. We hence formulate our thesis statement as follows:

Thesis Statement

Cloud consumers' advanced search queries can be used to infer new service requirements, such that cloud providers deliver solutions targeted at consumers' real needs.

1.3 Research Goal, Questions and Methodology

In this section, we introduce the research goal and from this we derive the related research questions. Further, we present the overall methodology used for this dissertation. The details on the concrete methods employed for answering the individual research questions are laid out in Chapters 2–5.

As motivated by the knowledge gap described in Section 1.2.4, our research goal is the following:

Thesis Goal

Develop a requirements elicitation approach that addresses the challenges posed by the cloud paradigm and supports cloud service providers in eliciting and understanding their consumers' requirements.

We adopted a pragmatic approach [Cre13] for our research since our main focus was to study *what* current problems cloud providers are facing regarding consumers' requirements communication and *how* these problems can be solved. Therefore, as advised by the pragmatic paradigm, we considered the thesis goal of central importance and applied the necessary techniques to understand the problem. Further, we chose data collection, analysis methods as well as technical solutions that were evaluated as most promising to provide insights into the problem [MK06]. Moreover, the pragmatic paradigm was deemed as appropriate also because our research is real-world practice oriented [ESSD08].

As far as the research methodology is concerned, we used an approach inspired by Wieringa and Heerkens [WH06] to meet the goal of this thesis. Our methodology has an iterative character, as shown in Figure 1.2. The cycle contains four major steps: problem investigation, solution design and evaluation, solution implementation, and implementation evaluation. We started our research with the problem investigation, as suggested by Mackenzie and Knipe [MK06]. Then, based on the requirements we gathered, we designed the solution and evaluated the concept. Once the design and concept were validated with cloud provider companies, we

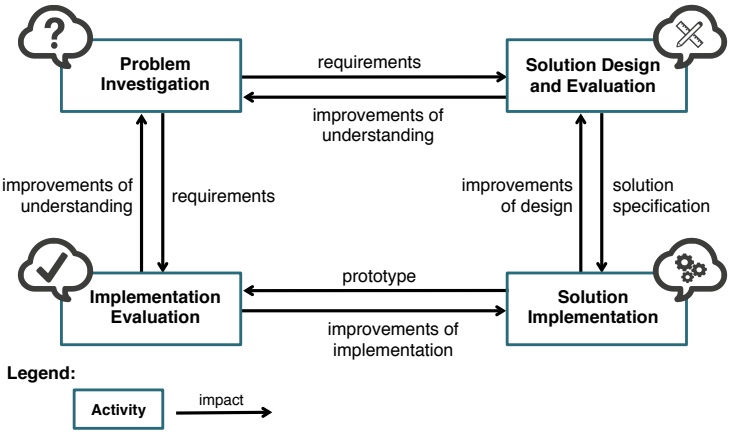

Figure 1.2: Our research methodology, adapted after [WMMR06] and [WH06].

implemented the solution and the resulting prototype was then evaluated with real-world and self-generated data. Naturally, we often had iterations for the four major methodology steps and for subsets thereof. For instance, once we collected requirements from cloud provider companies for the new dedicated cloud elicitation method, we designed the solution. However, the design was further enhanced later, upon receiving feedback on our ideas from cloud providers. All the other steps were treated in a similar fashion.

Therefore, to support problem investigation, our first research question is an exploratory knowledge question, belonging to the category of descriptive-process questions according to Easterbrook et al. [ESSD08]:

Research Question 1 (RQ1)

How do cloud service providers elicit consumer requirements?

To deliver an effective cloud requirements elicitation method, it is essential to first investigate how cloud service providers currently get to know their (potential) consumers and their requirements, what tools and methods they employ and what challenges they encounter. Once we know what the current state of practice is, we can design a method well tailored to their concrete needs. Thus the next two research questions are design questions [ESSD08].

Research Question 2 (RQ2)

What features should a dedicated cloud requirements elicitation method have to help cloud providers understand their (potential) consumers' needs?

Based on the results of the exploratory question RQ1, the necessary features for a dedicated cloud requirements elicitation method are defined. Having a concrete list of requirements that need to be satisfied, we can design the needed approach.

Cloud consumers often perform advanced online searches before they decide which provider and solution to choose. We therefore study how such consumer-generated data can be aggregated, modeled and analyzed such that new service requirements are inferred.

Research Question 3 (RQ3)

How can (potential) consumers' advanced search queries for cloud services be used to infer requirements?

Finally, we evaluate how our approach meets the thesis goal and requirements identified with RQ2 for a dedicated cloud elicitation method. For this, we use the following evaluation question that supports the implementation evaluation step in our methodology depicted in Figure 1.2.

Research Question 4 (RQ4)

How does our approach meet cloud service providers' needs for a dedicated cloud requirements elicitation method?

We consider our research successful if (1) a proof of concept of our approach is achieved, that solves the issues identified with RQ1 and (2) the results of the evaluation show that the requirements determined with RQ2 are met.

The next section gives an overview of our approach, which addresses the first three research questions.

1.4 StakeCloud in a Nutshell

To meet the goal of this thesis, we devised an approach called StakeCloud that supports cloud service providers in getting to

know their (potential) consumers and eliciting their requirements. The StakeCloud name stands for both the conceptual approach and its proof of concept implementation. In this section, we firstly introduce the general operation context of our method and then present a sample usage scenario.

1.4.1 Operation Context

The number of available cloud services has steadily grown recently and so has the number of consumers, whose needs are increasingly sophisticated. As a result, means for choosing the most appropriate cloud solutions are needed, given the indisputable paradox of choice [Sch04] consumers are facing. Therefore, as explained in Section 1.2.1, marketplaces have emerged as central websites which aggregate services from various providers that consumers can choose from. For instance, Intel Inc. launched the Intel Cloud Finder platform [Int15] that allows consumers to search for cloud service providers upon specifying the features their desirable solution should meet. Similarly, Cloudorado [Clo15] and Deutsche Börse Cloud Exchange [Deu15] enable (potential) consumers to make advanced searches that specify cloud service features at different levels, to identify the best matching cloud server, storage or hosting solutions. Moreover, it is forecast that this cloud marketplace model will continue spreading and gaining popularity in the near future [Dia14].

Irrespective of the marketplace, the working mechanism is the same for all existing such platforms, and is illustrated in Figure 1.3.

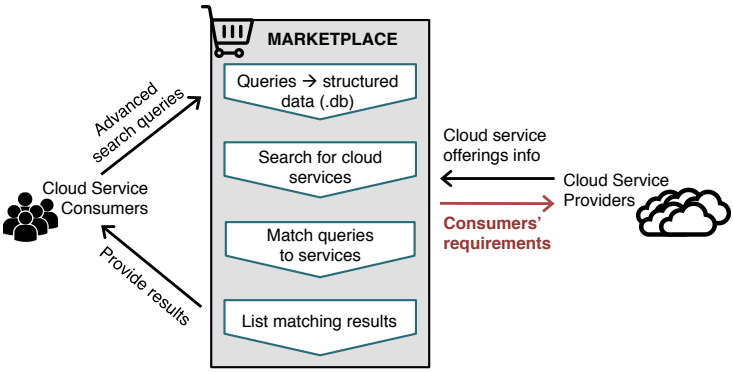

Figure 1.3: Marketplace working mechanism, adapted after [Tod12].

On the one hand, service providers input their cloud offerings into the marketplace, such that they become visible to and reachable for large populations. On the other hand, (potential) cloud service consumers access marketplaces and perform advanced search queries to find services that best match their needs. For this, they usually select their desired features from drop-down menus, using radio buttons or select intervals on predefined scales. At the moment, according to our knowledge, none of the existing marketplaces allows natural language text as input. As a next step, consumers' needs are saved by the marketplace in a structured manner, usually in the form of database entries (.db files). Based on the queries, a search among the existing aggregated cloud services is performed, to identify matching services. Then, a list of matching or partially matching cloud solutions is displayed in the consumers' browser, usually including a direct link to the original website of the cloud provider for each offering. From this point on,

based on the suggestions made by the marketplace, the consumer chooses the provider and starts the actual purchase directly with the service company.

This way, consumers' search queries are stored by the marketplace, but such data have only been used for marketing purposes so far, e.g., to recommend similar services [RV97]. The potential to model and analyze these data to infer new cloud service requirements has never been exploited so far. Therefore, our idea is to use consumers' advanced search queries to deduce new requirements. These requirements which are exclusively based on real search data can then be used by cloud provider companies to enhance and tailor their offering to better satisfy their customers (this activity is illustrated by the dark red arrow in Figure 1.3).

Our StakeCloud approach takes advanced search queries collected on marketplaces as input, models them and supplies complex means for their analysis to cloud providers. The final goal is to find requirements and combinations of features that eventually lead to developing new cloud services and novel classes of cloud solutions well targeted at consumers' needs.

Upon modeling the advanced search queries as fuzzy vectors, i.e. vectors whose elements are numerals in the range [0,1], and computing individual query frequencies, fuzzy Galois lattices are generated based on the input data. Galois lattices are mathematically equivalent to directed acyclic graphs with exactly one source node (the lattice supremum) and one sink node (the lattice infimum). Therefore, they can be represented graphically as directed acyclic graphs.

In our applied case of fuzzy Galois lattices, the nodes in the first level of the hierarchy correspond to the initial queries or clusters thereof. The lattice supremum represents a service that would satisfy all initial queries; however, implementing such a service in practice is virtually impossible or unfeasible, both technically and economically, in most cases. Therefore, the StakeCloud approach analyzes the lattice elements in the other hierarchy levels (lattice infima elements), which are compromise services that will satisfy the initial queries to limited extents. Please note that the visual representations of the lattices used as examples throughout this thesis may miss some of the edges due to a limitation of the graph representation library used for the implementation (vis.js). This cannot display edges that cross hierarchical levels in the lattice and limits itself to only showing the connections between adjacent levels of the hierarchy. However, this is only a visualization limitation and it does not impact the way the suprema and infima elements are computed and analyzed, thus having no influence on the outcome of the approach.

The StakeCloud method includes both modeling and analysis mechanisms that enable cloud providers to choose from the candidate compromise services, such that they can satisfy large populations of consumers with a minimum set of requirements implemented. Furthermore, our approach allows them to focus on particular features that are significant for their businesses or on key consumers. In addition, cloud provider representatives can model existing service packages and analyze how they can adjust their offering to better tailor it to the market needs and satisfy larger populations. In the following, we show how StakeCloud works in practice

and focus on its practical use. For more details on the meaning and construction of the fuzzy Galois lattices within our approach, further technical and theoretical specific aspects, please refer to [TG14] and [TKG15a], or Chapters 3 and 5, respectively.

1.4.2 Scenario

This section presents a sample scenario walkthrough for the tool-supported StakeCloud approach. This runs from the moment a dataset containing advanced search queries for cloud services is loaded until a preliminary decision is made for satisfying several queries selected by the user of the tool. In this context, the user of our approach is a representative of a cloud provider company, who is responsible for the features included in cloud offerings, e.g., a product manager. For this example, we chose a dataset composed of eight initial queries for simplicity reasons. However, our tool has been tested on datasets of thousands of queries, producing good results (e.g., it generates clusters for 1000 queries in less than 1.3 seconds). The scenario presented here is also shown in our one minute technical summary *video* that can be found at: http://goo.gl/qv5I5s.

We assume a cloud data storage company is interested in releasing a new service package for private users, that focuses on the features "higher storage space" and "improved uptime", labeled as features f9 and f10, respectively, in our dataset. Jo, the product manager responsible for this new service release, uses the tool-supported StakeCloud approach to understand what (potential) consumers'

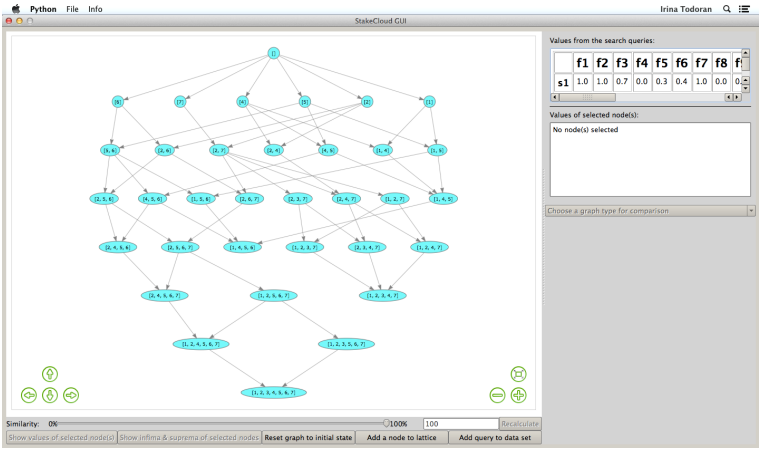

Figure 1.4: Scenario step 1: Dataset loaded and lattice generated.

requirements are and investigate what services would satisfy them. For this, he takes the following steps.

Step 1. Jo loads a dataset of queries collected via a marketplace. Each query is represented by a set of values that define ten different predefined features: type of consumer the service is aimed at, i.e. private (f1) or business consumer (f2), storage (f3), mobile support (f4), file recovery (f5), reliability (f6), AES (f7) and SSL encryption (f8), maximum size/file (f9) and uptime (f10). The lattice is automatically generated and displayed in the main panel of the StakeCloud Tool window (Figure 1.4). For more details on the tool characteristics and implementation, please refer to [TKG15b].

Step 2. He feels the lattice contains too many candidate services, so he decides to apply clustering. He sets the similarity degree

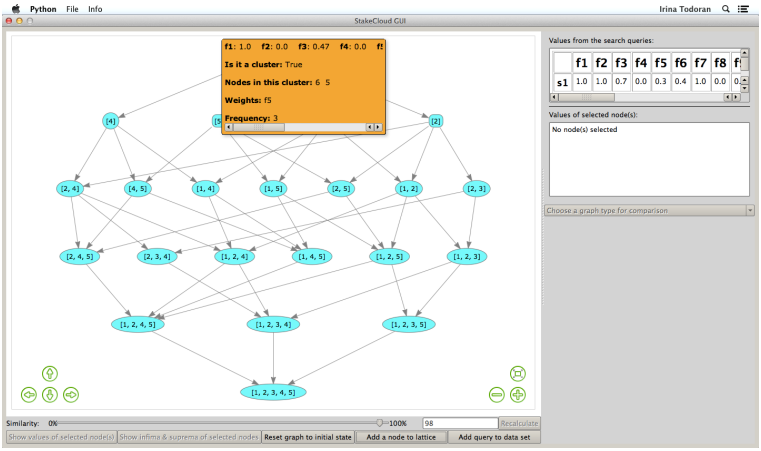

Figure 1.5: Scenario step 2: Clustering applied (similarity degree: 98%).

from the default value of 100% to 98% and clicks the "Recalculate" button. A similarity degree of 100% means that all the initial queries included in a cluster are 100% similar, i.e. identical. Therefore, when a dataset includes only unique advanced search queries, the clusters computed for a similarity degree of 100% coincide with the initial queries. A similarity degree of 0% means the queries included in the clusters are not similar at all, thus leading to one single cluster that includes all the queries in the dataset. In general, we define the similarity degree as a measure of the degree to which queries in a dataset cluster together: a value of n% means that all the initial queries included in a cluster are n% similar.

In the case of this scenario, the similarity degree of 98% means the queries in the clusters that are formed are 98% similar to each

Table 1.1: Advanced search query *[4]* and cluster *[5]*.

	f1	**f2**	**f3** GB	**f4**	**f5** days	**f6** %	**f7**	**f8**	**f9** GB	**f10** %
[4]	✓	✗	1024	✗	90	93	✓	✓	80	97
[5]	✓	✗	470	✗	30	97.3	✗	✗	60	100

other, as computed by our algorithm (for details on the clustering algorithm, please refer to Section 5.2.3). Similar nodes are automatically bundled and the resulting lattice is displayed, including clusters *[2]* and *[5]*. These are drawn as rounded rectangles (Figure 1.5). By hovering over them, Jo can see various details, such as the initial queries included.

Step 3. Now, Jo would like to see how his company could satisfy query *[4]* and the queries composing cluster *[5]* illustrated in Table 1.1, since these represent key stakeholders. The fuzzy vector representing query *[4]* describes a service intended for private users, that allows up to 1TB (1024GB) storage space, does not necessarily have mobile support, allows file recovery for files not older than 90 days, the reliability is minimum 93 per cent, the service ensures AES and SSL encryption, the maximum size/file is 80GB and the uptime is at least 97 per cent. The representative vector for cluster *[5]* describes a storage service for private users, that allows up to 470GB storage, does not necessarily have mobile support, AES and SSL encryption, files should be recoverable when not older than 30 days, reliability has to be at least 97.3 per cent, the maximum size/file allowed has to be minimum 60GB and the uptime 100 per cent.

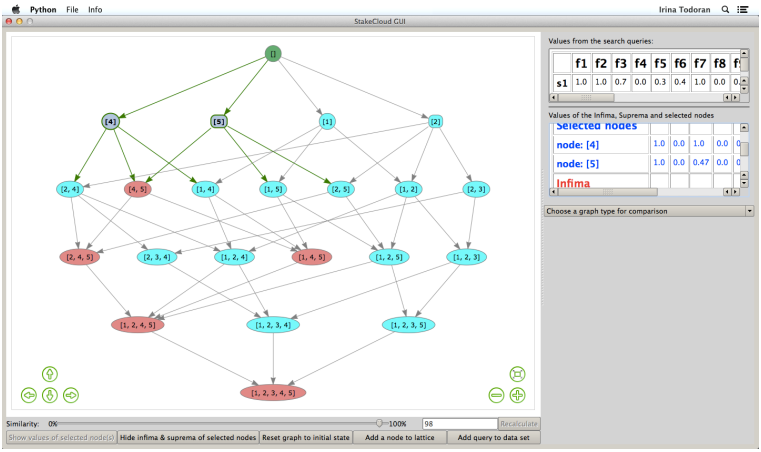

Figure 1.6: Scenario step 3: Nodes selected, and infima and suprema highlighted.

To perform the analysis, Jo selects the two lattice elements *[4]* and *[5]* and clicks the "Show infima & suprema of selected nodes" button. The supremum *[]* is immediately highlighted in green and infima elements in red (Figure 1.6). Moreover, the values of the corresponding fuzzy vectors for the highlighted elements are displayed in the middle right panel.

Step 4. The infima elements are candidate services for future implementation since they satisfy the selected queries to some extent. But which of them match the goals for the next release best? Firstly, Jo would like to know how many features they satisfy fully. Therefore, he chooses the "Suprema & Infima Analysis of Chosen Queries" from the dropdown menu, selects *[4]* and *[5]* and clicks "Show analysis". The graph displayed (Figure 1.7) shows

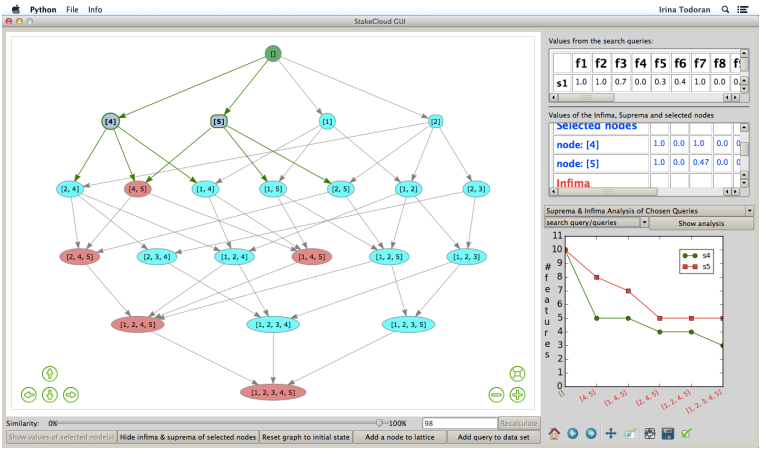

Figure 1.7: Scenario step 4: Suprema and infima analysis.

that, e.g., *[2,4,5]* and *[1,2,4,5]* fully satisfy an equal number of features, but Jo cannot tell from this graph to what extent features f9 and f10 are satisfied. These features were defined as critical for his business, as specified at the beginning of the scenario.

Step 5. To find out whether *[2,4,5]* or *[1,2,4,5]* better matches his company's target for the new service with regard to f9 and f10, Jo firstly selects the two candidate elements and then chooses "Feature Satisfaction Analysis of a Chosen Query" from the dropdown menu to study how f9 and f10 are satisfied by the two infima candidates relative to *[4]* and *[5]*. The graph for *[5]* in Figure 1.8 shows that, whereas f10 is satisfied equally well by both infima considered, f9 is satisfied 30% better by *[2,4,5]*. The same observation applies for *[4]*. Therefore, *[2,4,5]* represents a preliminary requirement that would satisfy well both *[4]* and *[5]*, thus meeting the company

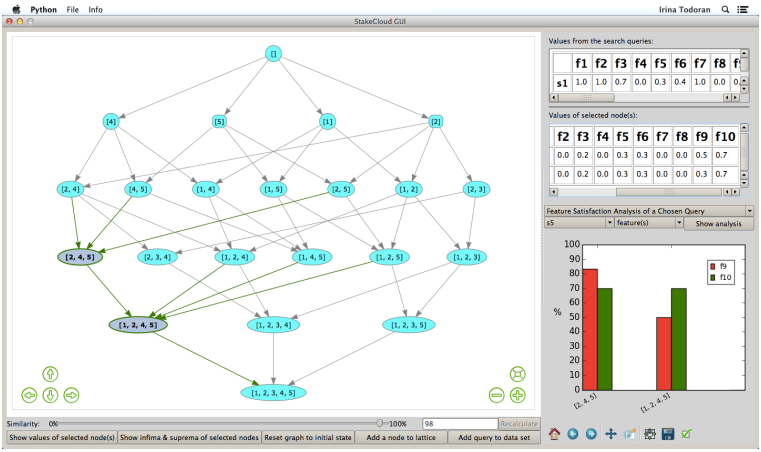

Figure 1.8: Scenario step 5: Feature satisfaction of *[5]* by the two infima.

constraints and goal for the next service release. Moreover, it also partially satisfies query *[2]*. Therefore, Jo managed to find a new class of services, *[2,4,5]*, that would satisfy his initial selection of queries: *[4]* and cluster *[5]*. This new class of services represents a compromise service in that it only partially satisfies some of the features requested in the initial queries, while still satisfying some features to a full extent.

For instance, the maximum size/file allowed by *[2,4,5]* is 50GB, which is slightly lower than the initially requested 80GB (by *[4]*) and 60GB (by cluster *[5]*). Another example is the file recovery compromise: files can be recovered when not older than 30 days, exactly as requested by cluster *[5]*, but this is a compromise when satisfying consumer *[4]* who requested a minimum of 90

days. Nevertheless, this new service candidate described by the new combination of requirements identified with the StakeCloud approach will satisfy a larger population, which is the trade-off for only partially taking into consideration some consumers' needs. When working with larger datasets, the provider will naturally aim at satisfying more queries, thus intensifying the analysis, but the incremental reasoning will be similar to the one presented in this sample scenario.

1.5 Roadmap and Chapter Summary

This dissertation is cumulative and its core consists of four standalone scientific peer-reviewed and published articles which are presented in Chapters 2–5, followed by the Conclusions. In this section, we outline these works and, for each of them, we (1) describe the motivation, (2) summarize their contribution, and (3) explain what research questions defined in Section 1.3 they answer. Figure 1.9 illustrates the roadmap of the thesis, including the relations between the four methodology steps, research questions and chapters.

Chapter 2: How Cloud Providers Elicit Consumer Requirements

Motivation. The existing state of the art presented in academic publications proved to be rather limited in describing concrete

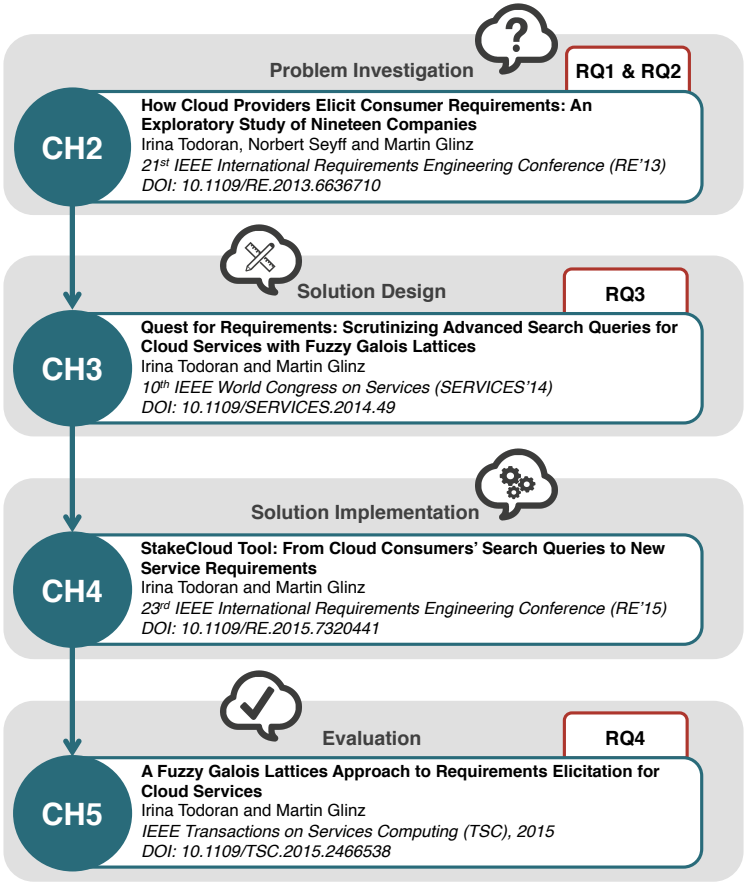

Figure 1.9: Thesis roadmap showing the relation between the chapters, publications, research methodology steps and research questions.

ways in which the requirements elicitation activity differs in cloud settings compared to traditional settings. It only outlined a few challenges encountered by cloud service providers and suggested some research directions instead of supplying methods that objectively support cloud providers in understanding their consumers' needs. Moreover, it was not clear what kind of approaches cloud providers use to elicit requirements.

Contribution. Chapter 2 is the foundation of the thesis since it represents the *problem investigation* step in our methodology. We conducted an exploratory interview study with 19 cloud providers to gain an in-depth understanding of how they perform requirements elicitation. The main contribution of this work lies in revealing what elicitation methods are used by cloud providers and clarifying the challenges related to requirements elicitation posed by the cloud paradigm. Further, we identified key features that cloud-specific elicitation methods should have.

Research Questions. This chapter answers the first two research questions.

Chapter 3: Scrutinizing Advanced Search Queries for Cloud Services with Fuzzy Galois Lattices

Motivation. Cloud challenges such as heterogeneous and globally distributed users, volatile requirements and frequent change requests cannot always be satisfied by existing methods. Therefore, a new dedicated cloud elicitation method is needed.

Contribution. Since consumers usually search online before they purchase a cloud solution, we propose analyzing their advanced search queries to infer new requirements. This chapter represents the *solution design and evaluation* step in the methodology. Our StakeCloud approach builds fuzzy Galois lattices for the terms that compose advanced search queries, thus enabling a thorough analysis of stored search data. This can support cloud providers in observing requirements clusters and new classes of cloud services, identifying the threshold for achieving satisfied consumers with a minimal set of requirements implemented, and further design novel solutions based on market trends. Moreover, the Galois lattices approach enables large-scale consumers' involvement and ensures the elicitation of real requirements unobtrusively.

Research Questions. Chapter 3 addresses the third research question by explaining how consumers' advanced search queries can be modeled and analyzed to infer new requirements.

Chapter 4: StakeCloud Tool: From Cloud Consumers' Search Queries to New Service Requirements

Motivation. Chapter 3 introduced the solution based on fuzzy Galois lattices on a conceptual level and provided a running example. Nevertheless, to evaluate the success of an approach in practice and follow the pragmatic paradigm of concretely solving a real world problem, the solution has to be tool-supported.

Contribution. This chapter represents the *solution implementation* stage in the methodology and describes the StakeCloud Tool which implements our approach. The StakeCloud Tool automatically builds fuzzy Galois lattices from given advanced search queries and provides requirements analysts with extensive clustering and analysis capabilities, as well as means for comparing different newly generated classes of services. The main contribution lies in a working prototype that has all the features described in Chapter 3 and beyond.

Research Questions. Since this methodology stage has a strong engineering focus and is not a research task, there are no research questions associated.

Chapter 5: Evaluating the Fuzzy Galois Lattices Approach to Requirements Elicitation for Cloud Services

Motivation. No solution is valuable unless it proves it can solve the problem it was designed for. Thus, an evaluation was needed to demonstrate how the fuzzy Galois lattices approach for cloud requirements elicitation we designed and implemented supports providers in gathering and understanding consumers' needs.

Contribution. Chapter 5 represents the *implementation evaluation* stage in the methodology. It extends Chapter 3 with the following. Firstly, the preliminary algorithm is enhanced by improving its performance and this functionality is added to the

StakeCloud Tool. Secondly, a similarity classifier is added, which allows flexible clustering of similar queries, thus improving the overall scalability. Thirdly, a series of experimental evaluations are conducted to verify our new requirements elicitation approach and explain how it meets the requirements outlined in Chapter 2. This chapter concludes that StakeCloud is a dedicated cloud elicitation method which operates with data that already exists, without the active participation of consumers and RE specialists.

Research Questions. Chapter 5 answers the evaluation research question RQ4.

1.6 Contributions

The contributions of this thesis are twofold.

Firstly, we analyzed the *state of practice with regard to requirements communication in cloud settings.* We revealed what methods cloud service providers currently use to understand their consumers' needs, identified the main elicitation challenges they encounter and determined what key features a dedicated cloud requirements communication approach should have.

Secondly, based on the lessons learned from the exploratory study with cloud provider companies, we developed *StakeCloud: a new dedicated cloud requirements communication approach.* The main components of StakeCloud are the following.

(i) A conceptual solution for analyzing cloud consumers' advanced search queries to infer new service requirements. This solution has its roots in Galois theory, but goes beyond this by extending the application to the fuzzy domain, enhancing it with clustering and analysis capabilities, and applying it in a field where it had not been employed before.

(ii) A practical implementation of the conceptual solution represented by the prototype StakeCloud Tool.

(iii) An evaluation of the implemented solution which demonstrates how the StakeCloud approach satisfies the requirements identified during the exploratory study and how it enables cloud service providers to elicit and analyze (potential) consumers' needs.

Chapter 2

How Cloud Providers Elicit Consumer Requirements

Original publication:
How Cloud Providers Elicit Consumer Requirements: An Exploratory Study of Nineteen Companies
Irina Todoran, Norbert Seyff and Martin Glinz
21st IEEE International Requirements Engineering Conference (RE'13)

Abstract

Requirements elicitation is widely seen as a crucial step towards delivering successful software. In the context of emerging cloud systems, the question is whether and how the elicitation process differs from that used for traditional systems, and if the current methods suffice. We interviewed 19 cloud providers to gain an in-depth understanding of the state of practice with regard to the

adoption and implementation of existing elicitation methods. The results of this exploratory study show that, whereas a few cloud providers try to implement and adapt traditional methods, the large majority uses ad-hoc approaches for identifying consumer needs. There are various causes for this situation, ranging from consumer reachability issues and previous failed attempts, to a complete lack of development strategy. The study suggests that only a small number of the current techniques can be applied successfully in cloud systems, hence showing a need to research new ways of supporting cloud providers. The main contribution of this work lies in revealing what elicitation methods are used by cloud providers and clarifying the challenges related to requirements elicitation posed by the cloud paradigm. Further, we identify some key features for cloud-specific elicitation methods.

2.1 Introduction

Requirements elicitation is a core activity in any requirements engineering (RE) process [SS97]. Using elicitation techniques that do not fit the characteristics of the project at hand increases RE costs and makes the project failure-prone [Tuu03]. Hence, numerous elicitation techniques, as well as methods for selecting the right techniques for a given project, have been developed and applied in practice [ZC05].

In the context of emerging cloud systems, providers of cloud services need to elicit the requirements of potential and actual service consumers in order to develop commercially successful services. However, the existing body of knowledge that cloud practitioners can rely on mostly consists of the well-known requirements elicitation techniques that have been developed for use in traditional system development settings [SS97]. Also, findings from the fields of market-driven and distributed RE [KDNoD+03, DZ03] are not directly applicable, due to the differences between the mass-market and the cloud computing domains. Moreover, no empirical evidence on the elicitation methods utilized by cloud providers is available.

Therefore, we designed an exploratory study to better understand the state of practice in industry and the degree to which existing research results support cloud providers' needs with regard to requirements elicitation. Moreover, where applicable, this study investigates how traditional and market-driven methods are used,

to what extent, where they are adapted to better suit individual needs, and where new, ad-hoc approaches are chosen. Since most of the participating companies were not cloud providers from the outset and only later adopted the cloud model, we also analyze the impact of their evolution on the elicitation process. The study consists of in-depth semi-structured interviews with 24 respondents from 19 companies located in 10 different countries.

The paper is organized as follows. We clarify the terminology and particular cloud computing features in Section 2.2. Section 2.3 introduces the research methodology, including research questions, study design, and threats to validity. The key findings are presented in Section 2.4 and then summarized and discussed in Section 2.5. Section 2.6 presents related work, and the last section concludes the paper with a summary and outlook.

2.2 Cloud Computing Particular Features

2.2.1 Stakeholders in the Cloud

In a traditional setting, clients typically run the systems at their own premises, either owning and maintaining the software themselves or owning licenses to run the software or parts thereof. Suppliers, on the other hand, sell or license, install the systems and potentially provide maintenance and consulting.

In the cloud context, consumers do not own solutions any more, but subscribe to services which they can use on demand subsequently. Cloud services are offered by cloud providers. A *cloud (service) provider* is an organization, rarely a person, responsible for making a service available to interested parties. According to the US National Institute of Standards and Technology (NIST), a cloud provider "acquires and manages the computing infrastructure required for providing the services, runs the cloud software that provides the services, and makes arrangement to deliver the cloud services to the cloud consumers through network access" [LTM+, p. 7]. Therefore, the provider is the actual owner of the solution [LBRK10].

A *cloud (service) consumer* is the stakeholder that uses the cloud services, and is represented by "a person or organization that maintains a business relationship with, and uses the service from a cloud provider" [LTM+, p. 5]. Consequently, both Business-to-Business (B2B) and Business-to-Consumer (B2C) models are supported.

A *cloud system* is a system where computing resources are provided on demand, as services, through network access, and the main stakeholders are the cloud consumers and providers, with the characteristics described above. The service can most often be Software as a Service (SaaS), Platform as a Service (PaaS) or Infrastructure as a Service (IaaS) [LBRK10]. Consumers utilize services delivered by a provider based on a trust agreement (most frequently in the form of a Service Level Agreement).

2.2.2 Cloud versus Conventional, Mass Market Systems

The cloud computing setting is characterized by a large number of heterogeneous, globally distributed consumers, which can go beyond what traditional requirements elicitation methods are able to support [LRA02, Saw00]. Moreover, change requests are frequent, resulting in unstable and volatile requirements [Her07]. Some of these characteristics are similar to those of international mass market settings, with diverse and often remotely located consumers. However, from a requirements elicitation perspective, the findings from the market-driven RE field are not generally applicable in a cloud context.

In the market-driven domain, many products have some local market, such that some of the stakeholders are within reach and available for direct interaction. In these cases, traditional requirements elicitation methods can be applied at least to some extent. Also, product managers use their own professional experience to invent requirements [Pot95] when relevant stakeholders cannot be reached. As an alternative, some companies perform market studies to understand the trends in consumer requirements, which often leads to imitating what competitors do. Other organizations conduct in-depth studies and test their prototypes with selected stakeholders, and then generalize the results to the mass market.

Cloud systems differ from mass market systems in that cloud services usually do not have a local, individually reachable and

easily exploitable consumer base to involve in the elicitation process. Furthermore, inventing requirements or imitating competitors are often seen as unreliable approaches due to the young age of the cloud model. Thus, despite being relevant for the cloud, the existing methods from the mass market domain do not suffice for cloud providers' needs. This motivates us to focus on the challenges they face in requirements acquisition, given the importance of the elicitation process.

2.3 Research Methodology

To understand the current state of practice of requirements elicitation for cloud systems, we conducted an exploratory study with 19 cloud provider companies. We chose a qualitative research approach [Rob02] since such an approach focuses on information depth and allows for investigating diverse and complex data [Bla10]. We used semi-structured interviews based on a pre-defined interview instrument[1]. All questions were elaborated to support three research questions (RQs; see the section below), and served as a starting point and structure for discussion. Since the interviews were semi-structured, the interviewer also had the flexibility to adapt according to individual circumstances, focus more on specific areas or discard questions which did not apply. The interview instrument included five parts. The first questions focused on the characterization of the company and interviewee. The next two parts included questions on cloud providers' methods for reaching

[1] http://www.ifi.uzh.ch/rerg/people/todoran/Interview_Instrument.pdf

consumers and identifying their requirements. Then, the state of practice of the company was discussed in comparison to competitors, and the interview was closed by analyzing elicitation-related challenges and possible mitigation plans.

2.3.1 Research Questions

RQ 1: What methods do cloud providers use to elicit consumer requirements?

Firstly, we are interested in finding out what requirements elicitation methods cloud providers currently use, if these are well-known RE methods, adapted traditional methods or simply ad-hoc approaches. Additionally, we analyzed the criteria used for method selection.

RQ 2: How do cloud providers' needs for elicitation methods differ from traditional software/hardware providers'?

Secondly, we investigate how cloud providers are different from traditional software or hardware providers as far as elicitation methods are concerned, what causes the potential differences, and if these have any impact on the elicitation method selection. For this, we focused on the companies which shifted from the traditional model to cloud computing.

RQ 3: To what extent can the existing elicitation methods satisfy cloud providers' needs?

Thirdly, we analyze if the existing methods suffice for cloud providers, or they require dedicated approaches, specifically tailored to their needs.

2.3.2 Study Design

Initial Preparation

The interview instrument was first elaborated as a list of questions linked to the RQs and the goals of the study. Then, it was validated with a group of RE researchers, and further improved. As a next step, the interview was piloted with one researcher from University of Zurich and one practitioner, who had not been previously involved in the elaboration of the study. During the pilot interviews, possible misunderstandings of questions were identified, new related areas interesting for investigation were found and thus some new questions were added. Moreover, the time needed for each part of the interview was measured. We adjusted the interview instrument based on the lessons learned while performing the pilot interviews.

Selection of Participants and Demographics

The sampling strategy we used for selecting the participating companies was convenience sampling [Pat90] within our direct contacts network. We first contacted employees of cloud provider companies we personally knew. Then, during a short discussion via e-mail or Skype, we evaluated if their profile and position within the company fit with the needs of our study. If they fit,

they participated themselves; if they were not a good fit, they recommended other employees.

As the participants needed for the study had to have a good overview of the requirements elicitation process and related company needs, only software architects, division and project managers, consultants and software engineers who had direct contact with requirements identification activities were recruited.

In total, the study is based on 26 data points from 19 different companies, located in 10 countries. From 7 companies, we had two respondents, and from each of the other organizations we interviewed one representative.

When we discussed with two employees of the same company, the interviews were individual and usually targeted complementary topics. This strategy was chosen to ensure we have the best-matching professionals responding to each part of our interview.

All participating companies have been cloud providers for at least one year and a half, and active in various domains, as outlined by Table 2.1 (built according to the guidelines in [IG11]). Only three of the nineteen organizations were founded as dedicated *cloud providers*, i.e. they adopted the model of delivering services on demand through network access from the outset. The others started as traditional software or hardware providers and only later extended their offer to the cloud. In Table 2.1, we use the term *hybrid* to identify such providers that evolved from traditional software or hardware suppliers to cloud providers, and currently have

Table 2.1: Companies Overview.

Company	Type	Domain of Activity	Deployment Model
C1	hybrid	ERP systems	public (SaaS), private
C2	hybrid	Document Management systems	private
C3	hybrid	ERP, BI, SW architectures, advisory	public (IaaS, PaaS, SaaS), hybrid, private, community
C4	hybrid	Document Management systems, ERP, process planner, intranet	public (SaaS)
C5	hybrid	Dedicated servers, data centers, web hosting	public (IaaS, SaaS), private
C6	cloud	Video hosting and sharing, copyright video provider	public (SaaS)
C7	hybrid	Antivirus solutions	public (SaaS)
C8	hybrid	Dedicated servers	private
C9	hybrid	Infrastructure, hosting, consulting services, various software	public (IaaS, SaaS), private
C10	cloud	Predictive analysis, data management	public (SaaS)
C11	hybrid	Ticketing services	public (IaaS, SaaS), private
C12	hybrid	IT, business consulting, outsourcing	public (IaaS, SaaS), private
C13	hybrid	Customized software for bluetooth equipment	private
C14	hybrid	Multimedia and creativity software	public (SaaS)

Table 2.1: Companies Overview - continued.

Company	Type	Domain of Activity	Deployment Model
C15	cloud	Managed cloud servers, collocation, private cloud setups	public (IaaS, PaaS, SaaS), private
C16	hybrid	Software for aerospace, defense, transportation and security markets	private
C17	hybrid	Procurement software	public (SaaS)
C18	hybrid	Language translation services	private, hybrid (with public: SaaS)
C19	hybrid	Energy consumption measurement systems	private

all the characteristics of cloud providers, as defined in Section 2.2. As far as the deployment model is concerned, they provide public, private or both services. For the public model, the SaaS type seems to be more frequent in our dataset than PaaS or IaaS. For confidentiality reasons, we do not disclose further data about the companies.

To ensure that interviewees' profiles fit well with the purpose of our study and thus provide relevant information, we asked each of them to evaluate their own RE experience in years, and the results are shown in Figure 2.1(a). Almost half of the participants (46.2%) reported between 4 and 8 years of experience in the requirements engineering field, 23.1% reported between 2 and 3 years, and 15.4% stated they had minimum 9 to 13 years. Furthermore, 11.5% self-reported 14 to 17 years and 3.8% more than 18 years of practical

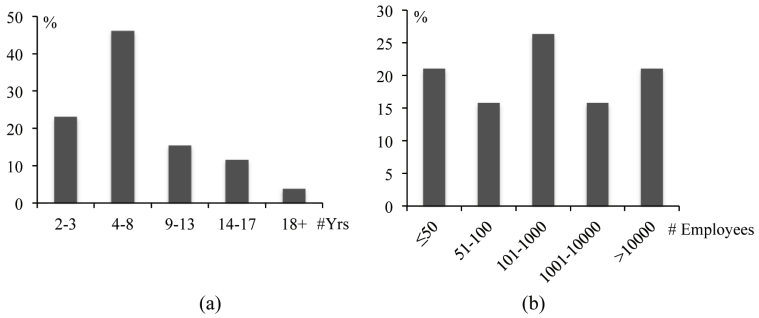

Figure 2.1: Distribution of interviewees by (a) years of RE experience and (b) size of participating companies.

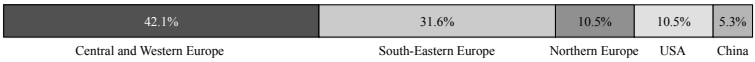

Figure 2.2: Geographical distribution of participating companies.

experience of RE. Since we did not involve anyone with less than 2 years of experience, and the mean was 6.9 years, we consider the sample relevant for the goal of this empirical study.

We did not include any age, gender or nationality assessment in our study, since we considered these would not have any impact on the participants' attitudes towards requirements elicitation activities in cloud systems. However, we noted that the majority was represented by males, and only 34.6% of the participants were females.

As far as the geographical distribution of the participating companies is concerned, we collected information from 10 different

countries, located in 5 geographical regions, as depicted in Figure 2.2. It can be noticed that the majority of responses (84.2%) came from Europe, and only 10.5% from the American continent (United States of America) and 5.3% from Asia (China). Within Europe, most of the respondents were employed by companies in Central or Western Europe.

With respect to size, our study covered all types of companies, from very small (under 50 employees) to very large (more than 10'000 employees). Figure 2.1(b) presents this distribution which, to our surprise, is a symmetrical graph. About a quarter (26.3%) of the organizations which took part in the interviews have between 101 and 1'000 employees, 21.1% have less than 50 and more than 10'000 employees, respectively, and 15.8% have between 51 and 100, and from 1'001 to 10'000 employees, respectively.

When asked whether the cloud services provided are developed internally, within the company, all respondents answered positively. Moreover, half of the companies also aggregate their services with other software from third parties, and 25% also resell or integrate complete cloud services from other providers, potentially including some customization.

Data Collection and Analysis

The interviews were conducted between November 2012 and January 2013, and their duration varied between 45 and 90 minutes,

with an average of 70 minutes. We conducted the interviews over Skype or Google Talk and, when possible, the video feature was enabled. In two situations, when neither Skype nor Google Talk were available for participants, the landline was used.

The data analysis consisted in first aggregating and structuring all the information collected, so potential patterns can be observed and statistical calculations can be performed. Then, the interesting aspects were selected and analyzed more thoroughly. Section 2.4 presents our key findings.

2.3.3 Threats to Validity

From the early stages of the study design, we considered the possible threats to validity for our results. As with any empirical study, it is difficult to completely resolve all the issues which can appear. In this section, we discuss the potential threats to validity identified, according to the categorization by Wohlin et al. [WRH+12].

Conclusion validity issues are caused by the inability to draw accurate conclusions based on the study. We attempted to alleviate the risk associated with measures reliability by conducting two pilot interviews prior to the study, to identify any possible misunderstandings of the questions. During all interviews, we always encouraged the interviewees to ask for clarification in case something was unclear. Moreover, we used redundant questions, to ensure the answers were consistent. However, we cannot claim

that no misunderstandings occurred at all. All interviews were conducted by the first author, thus avoiding discrepancies caused by differences between interviewers. We mitigated the problems of phone interviews by scheduling all meetings in advance and holding them as private meetings such that no third parties were disturbed or involved in the discussions. We also made sure that the interviews were not disturbed by poor connection quality. In the cases where two representatives from the same company were interviewed, interviewees were asked not to talk about the content of the interview before both interviews had been conducted.

Internal validity refers to the possible causal relationship between treatment and outcome. Our selection of participants was constrained to the cloud provider employees recruited by our contacts and, where applicable, our contacts themselves. Participation was completely voluntary and this can skew our results towards practitioners who were highly motivated and interested in the research topic. However, avoiding the self-selection principle is virtually impossible in this type of study. As far as maturation is concerned, we always documented ourselves about the organizations interviewed and the cloud services provided prior to interviews. The aim of this was to avoid tiredness and boredom during discussions. Additionally, interviews were restricted to maximum 90 minutes.

Construct validity is related to generalizing the results beyond the study. In this sense, we tried to eliminate mono-operation bias by aggregating data from various sources (public company information, employees). To avoid evaluation apprehension, participants

were assured that all the information they provide is anonymized and only used for research purposes, and we asked them to report if they feel uncomfortable about the discussion at any stage. Regarding the level of constructs, all participants were required to self-assess their RE experience. However, this threat cannot be completely dismissed. Hypothesis guessing is another construct validity threat which could have occurred, although the study description explained that our aim is to gain a deeper understanding of the phenomena, in an exploratory way, and did not suggest any expected results. As already mentioned, we also conducted two pilot studies and had the interview instrument checked by RE specialists to ensure the questions do not lead to any bias.

External validity threats limit the ability to generalize results to industrial practice. Since this is a qualitative study, it is rather difficult to generalize beyond the given settings and replicate the same contexts. However, we consider that the lessons learned from such a study are important and useful for industrial practice, and can be applied in other organizations as well. Therefore, we consider we can generalize our results, taking into account the sample size (19 companies) and the representativity of the participating providers: they cover all types of cloud services and all deployment models, from various domains of activity and different geographical locations. Around 80% of individual interview participants were recommended by our contact persons within cloud providers, which reduces the threat of interaction of selection and treatment (the rest of 20% were our direct contacts).

2.4 Key Findings

We assigned one investigation aspect to each of the research questions. For the three aspects, we present the key findings and elaborate on the corresponding evidence data. Due to space constraints, we focus only on the study results related to the three RQs.

2.4.1 Requirements Elicitation Techniques in Use - RQ 1

The core of the discussions we conducted with practitioners focused on the requirements elicitation techniques they use for identifying cloud consumers' needs. This section presents the aggregated key findings related to this investigation aspect. Our discussions included two categories of requirements elicitation methods: requirements elicited from existing service consumers, and requirements coming from potential consumers, i.e. who have not signed a Service Level Agreement (SLA) document with the provider up to the moment of the elicitation process.

Finding A.1. Traditional approaches (interviews, questionnaires, analysis of existing documentation, surveys) and prototyping are the most popular and highly applied existing requirements elicitation methods among cloud providers.

Evidence for Finding A.1. To steer the conversation and have a concrete and common starting point, we used an interview appendix[2] which listed the main elicitation techniques, according to Nuseibeh and Easterbrook [NE00]. For clarification, we reproduced the techniques in Table 2.2, keeping the same main categories used originally by the authors: traditional, model-driven, group techniques, cognitive, prototyping and contextual [NE00]. The main reason for using these approaches is that they represent a rather comprehensive list of well-known techniques, widely used as reference by the research community. We chose to treat brainstorming from two different perspectives: on the one hand, as a method used externally, to elicit requirements from consumers; on the other hand, as a method used in company internal meetings, involving only employees. This decision was made after noticing that our interviewees strongly differentiated between the two.

We designed two questions which share these same items, but have different scopes: "q_1: select the techniques you are familiar with and you previously used" versus "q_2: select the techniques your company uses in the elicitation process for cloud services".

This parallelism has two main advantages: consistency check and the possibility to calculate the importance rating, as outlined by Bettenburg et al. [BJS+08]. Regarding consistency, we considered that all techniques chosen as implemented should also be known to the respondent, since all interviewees self-reported to be involved in the requirements-elicitation or related processes. If reported otherwise, we asked again to make sure the question was understood

[2] http://www.ifi.uzh.ch/rerg/people/todoran/Appendix_Interview.pdf

Table 2.2: Elicitation Techniques, after [NE00].

	Traditional		**Model-driven**
A	Questionnaires	J	Scenarios, e.g. CREWS
B	Surveys	K	KAOS
C	Interviews	L	i*
D	Analysis of existing doc.		
	Group Elicitation		**Cognitive**
E	Brainstorming - externally	M	Protocol analysis
F	Brainstorming - internally	N	Laddering
G	Focus groups	O	Card sorting
H	RAD/JAD workshops	P	Repertory grids
I	Prototyping	Q	Contextual (ethnography)

correctly. Regarding importance of elicitation methods in cloud systems rating, we can infer the individual importance given to each technique by using the following formula:

$$\text{Importance}(i) = \frac{N_{1,2}(i)}{N_1(i)}$$

Here, for technique i, $N_1(i)$ is the number of responses in which the technique was selected in q_1. Similarly, $N_{1,2}(i)$ is the number of responses in which technique i was selected in both q_1 and q_2. The assessment of the importance rating is depicted by Figure 2.3. When importance tends to 1, then the technique is both known and widely implemented in practice. When it tends to 0, it is not used by practitioners, even if some may be familiar with it.

It can be observed that importance is highest for interviews (C), focus groups (G), analysis of existing documentation (D) and prototyping (I).

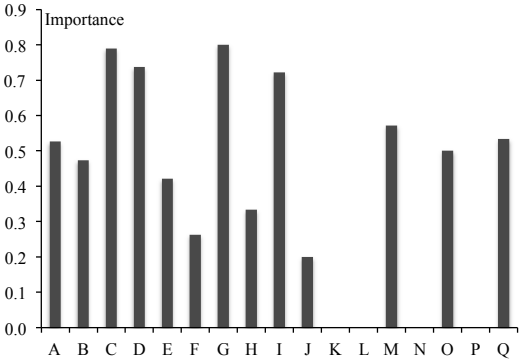

Figure 2.3: Importance rating of techniques, cf. Table 2.2.

Figure 2.4 depicts the general assessment of the 17 reference techniques. The horizontal axis shows the number of participating companies where at least one of the interviewees was familiar with the methods, and the vertical axis shows the number of organizations which successfully implement each method. This analysis generates some clusters, with different properties.

The top left corner of the graph is naturally empty, since the two questions were designed using the parallelism principle, thus allowing for consistency check. The bottom left corner contains techniques which are neither known, nor implemented in practice. Methods such as KAOS and i*, which receive significant attention in the RE research field, were only reported as known by one company representative, and were not implemented by any of the 19 participating organizations. Although familiar to more interviewees and implemented in up to four companies, card sorting, RAD/JAD workshops, scenarios and protocol analysis are still techniques

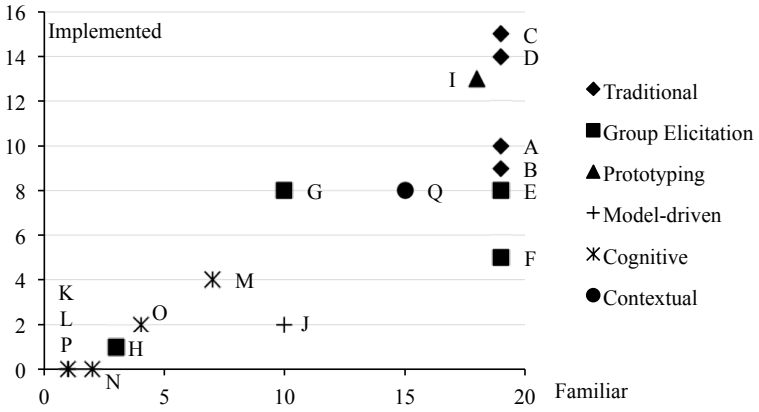

Figure 2.4: Familiar and implemented elicitation techniques.

with very low industry penetration and rather unpopular. It can be noticed that all the cognitive and model-driven methods are situated in this cluster.

The focus groups technique occupies a central position in the graph, being known to respondents from 10 companies, and applied successfully in 8 out of 19 organizations. The bottom right corner is populated only by the brainstorming method (internal and external) and ethnographic studies, showing that industry has knowledge about them, but does not implement them on a regular basis.

The top right corner cluster contains prototyping and all traditional methods. This means that these techniques are widely known to practitioners and commonly implemented as well. The top three are: interviews, analysis of existing documentation and

prototyping, which are also among the top rated as far as the calculated importance is concerned. This can also be read on the graph in Figure 2.4 importance is high for those techniques which are closest to the diagonal.

We did not limit our discussions to the interview appendix, but went beyond it to also discuss other methods which may not have been included in our list based on [NE00]. Respondents mentioned additional methods such as collecting user feedback, attending field conferences and other events, but none of these had an implementation frequency higher than 21%. Therefore, if we represented them in Figure 2.4, they would belong in the bottom left corner cluster, along with the other methods which are only known and applied by a small number of companies.

Finding A.2. Most of the existing methods are very difficult to nearly impossible to apply in the cloud context.

Evidence for Finding A.2. 63.2% of the interviewed organizations provide cloud services to consumers that are located in regions or even countries different from any of the physical company branches. This very distributed nature and sometimes impossibility to reach consumers for face-to-face meetings were reported by our respondents as significant challenges posed by the cloud, which hinder the implementation of existing methods. In spite of some differences between companies regarding preferred elicitation methods and application scenarios, there was one point the large majority (89.5%) agreed on: the difficulty of method implementation.

For instance, one interviewee from a European cloud provider explained: "We've been recently contacted by a company representative from New Zeeland who asked for one of our DMD [Document Management Systems] solutions. In general, before we customize and sell these solutions, we use extended workshops and observation at the client's premises to understand their requirements. I had to deny that request, we couldn't have afforded such a client from a different continent. It would have been impossible to understand what he wants." This is only one example of how two of the existing methods fail in practice in the cloud context.

One interviewee from an American B2B provider shared a different experience: "We use brainstorming a lot during the first phases of the [elicitation] process, but that gets tricky when we serve a company located too far. What we tried several times was to do it somehow online, using GoogleDocs. It works to some extent, but you can never tell what other things people do at the same time, you sometimes notice they are distracted". In this second example, a workaround for the problem was found and implemented, but the quality was reported as low, due to lack of control and impossibility to ensure optimal conditions for applying the technique.

Finding A.3. Almost all cloud providers (94.7%) use ad-hoc elicitation methods at some point during their process. Moreover, the general criterion for method selection is also ad-hoc.

Evidence for Finding A.3. With no exception, all our respondents consider the cloud field still young, and the large majority agrees that there is a lack of dedicated mature elicitation methods they could apply.

For example, 21% of the participating companies reported to heavily use their marketing human resources to perform market studies on their competitors; then, they try to imitate them in service offering. Two of these companies added that this "may not be a real elicitation method, but it helps". Similarly, 21% generalize what is known from existing consumers and assume that will also hold for potential future clients. Furthermore, 26.3% of the practitioners consider that internal company knowledge is the most important method for coming up with new requirements. In this case, the main source is the personal and professional experience of the employees. We asked these respondents about the reasoning behind this ad-hoc method, and the common argument was that "our people are good enough to know". Another interviewee expressed her disappointment: "there's not much else we can do". About 10% of the companies reported to rely on simply guessing and inventing requirements, but it was not clear if this is also based on internal knowledge, market monitoring or anything else.

As far as acquiring requirements from potential consumers is concerned, the providers which develop services for public institutions explained they get numerous hints from the self-defined requirements published by these organizations. However, they added that there are numerous situations when those requirements need to be refined, since they do not always express what is needed in reality. In these cases, further elicitation methods have to be used. Moreover, other companies organize or attend conferences and various fairs (15.8%) to bring potential consumers together and get in touch with them more easily.

Strongly related to the previous finding, the widely spread usage of ad-hoc methods was reported to be linked to the impossibility of applying existing techniques. 52.6% of the organizations even called the traditional methods “useless” or “old-fashioned”. The others refrained from using such strong words, but the general attitude towards the existing approaches was that they need to be either adapted or replaced by something new to meet the cloud particularities. Nevertheless, two companies reported that they are satisfied with what is available at the moment, since they do not feel there is a special need for dedicated methods for cloud providers.

According to Nuseibeh and Easterbrook, the criteria for choosing elicitation techniques should depend on time, resources available and type of information that needs to be elicited [NE00]. Our study shows that things are slightly different in practice. 52.6% of the companies interviewed agreed they use ad-hoc criteria for choosing elicitation methods, and that they do not follow any standardized processes. The argument given was, once again, the young age of the cloud paradigm. About a quarter (26.3%) emphasized the consumer reachability aspect in the cloud, which often determines the methods to be used - some may not apply at all if consumers are located remotely. Other criteria mentioned were the ease of use (10.5%) and the target group (5.3%). Only 15.8% of the participating companies choose the elicitation techniques based on available financial resources and 21% based on the type of information needed.

2.4.2 Cloud versus Traditional Providers' Methods - RQ 2

With the second investigation aspect, we aimed at understanding if cloud providers perceive any particular needs with regard to requirements elicitation, compared to their previous experience as traditional providers. Therefore, this discussion is limited to those companies which underwent this shift.

Finding B.1. The cloud calls for methods which fit for more heterogeneous audiences, take less time, and can be applied remotely.

Evidence for Finding B.1. 16 of the 19 companies interviewed developed from providing traditional solutions to supplying cloud services during the recent years. We investigated if they tried to apply the same elicitation strategies they used previously also in the cloud, or they started new with something different. A large majority (87.5%) agreed they tried to apply methods used before, and reported that 65% of the attempts failed. There are several causes for this situation.

In 62.5% of the cases, practitioners found it difficult to draw the profile of their average consumers, or did it in very general, vague terms, such as "individual consumers who need to host their data in the cloud" or "small and medium businesses (SMEs) which deal with large amounts of documents". It was generally admitted that the target audience for cloud services is much more heterogeneous than the audience for traditional software or hardware products.

Secondly, several respondents observed that services are developed and launched at a much faster pace in the cloud, and this also calls for suitable requirements elicitation methods: "we cannot afford to spend half a year to collect data from potential consumers with questionnaires when we must have the service on the market in two months". When asked if they found a way to address this issue, the answer was that they use agile development to have frequent releases. However, as far as requirements elicitation is concerned, no real solution which supports a shorter time to market is available. According to the interviewees, the cloud concept is too young and most cloud providers are still too inexperienced: "we all just try different things".

Thirdly, due to the distributed nature of the cloud previously discussed, cloud providers need elicitation methods which do not necessarily require physical meetings. One of the study participants explained: "we tried to somehow adapt well-known methods and, for example, turned our old workshops into online conferences - but this is not always easy, somebody's connection fails, another is not allowed to use a specific conference platform and so on". 60% of the practitioners who tried to adapt existing methods reported that this kind of failed attempts usually discourage future trials, so different methods are chosen in upcoming projects. For our study, this applies for both methods used in early elicitation phases and more in-depth techniques.

Finding B.2. Automation is needed to a larger extent for eliciting cloud consumer requirements.

Evidence for Finding B.2. Although we did not have any specifically designed questions for the automation topic, this recurrently occurred during our interviews with practitioners, especially when discussing about the limitations of the current methods and related challenges.

For instance, when asked if they found a workaround for the stakeholder identification and reachability problem, respondents' general answer was "no", "not really", or they suggested employing more human resources who can travel to meet consumers, which is expensive. However, two of the practitioners explained that they had some attempts to solve this at much lower costs, using automation. They measure usage points on the services provided, i.e. they monitor which features are utilized by consumers frequently and how, and which are only rarely employed. This is used on both final products and α- or β-releases. The information is automatically sent to the provider every week, and then processed by a product manager. As reported by one interviewee, "this is kind of indirect feedback for us, but it also brings new requirements sometimes, or at least some hints". He further explained that this method does not require heavy financial investments, and also has the advantage of being unobtrusive.

In line with this are also the comments from other practitioners: "ideally, we'd need something which helps us get needs from people everywhere, not only in [city name], but how do you get to them?" Here, the geographical coverage was strongly emphasized. According to another respondent, "costs are very important for us, so anything which requires minimum costs and automatically brings requirements would be highly desirable".

2.4.3 Do the Existing Methods Suffice? - RQ 3

To answer the third research question, we looked into the details of the providers' development history and asked them to evaluate themselves and their competitors with regard to existing best practices adoption. The main findings show that the current methods support providers' needs only to a limited extent, often leading to general dissatisfaction.

Finding C.1. Cloud providers' satisfaction level with regard to implementing existing elicitation approaches is low to medium.

Evidence for Finding C.1. We asked the study participants if they are happy with the implementation level of requirements elicitation best practices in their companies. For this, we told them to use a 5-point Likert scale [Lik32], where 1=strongly disagree, i.e. the company implemented nearly nothing, and 5=strongly agree, i.e. the company uses existing practices successfully, on a regular basis.

Moreover, we asked them to evaluate their companies in comparison to their competitors, using a similar 5-point Likert scale. The results are shown in the boxplots in Figure 2.5. Value 0 on the vertical axis was reserved for answers which did not include a Likert score (when participants could not provide an answer).

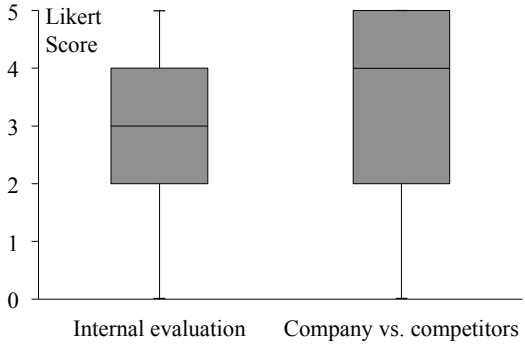

Figure 2.5: Evaluation of cloud providers' satisfaction.

The left boxplot represents the distribution for the internal cloud providers evaluation on the adoption and implementation of elicitation techniques. More than half of the interviewed cloud providers (68.4%) evaluated themselves as levels 2 or 3. Our respondents generally placed their employers under or around the average range, and only 10.5% of the organizations considered they were doing very well regarding methods adoption and implementation in practice. As a result, the median is 3 points, and the representation spans the quartiles Q_1=2 and Q_2=4, as depicted in the graph.

As far as the comparison to competitors is concerned, most cloud providers evaluated themselves to be better than competition at successfully applying existing techniques. The right boxplot in Figure 2.5 shows that, in this case, the calculated median is 4, which highlights that participants had a rather positive attitude towards their companies when comparing them to competitors.

Most considered their employers leaders on the market regarding elicitation methods used, thus generating a representation between quartiles Q_1=2 and Q_2=5.

This analysis shows that cloud providers generally evaluate themselves as average or under average in elicitation methods usage, but consider they are still better or much better than their competitors. This means that cloud providers' general satisfaction level with regard to implementing existing elicitation approaches is low to medium.

Finding C.2. In more than half of the cases, public cloud providers' dissatisfaction is caused by their evolution history.

Evidence for Finding C.2. We differentiate between several categories of public cloud providers, based on the type of public cloud they supply: IaaS, PaaS, SaaS or others. All our interviews included one question about the evolution history of the company: if they started as a cloud or as a traditional provider, and how this shift happened. In this analysis, the companies which started their business as cloud providers are excluded, since they did not go through a delivery shift (traditional to cloud provider) in their history. Therefore, among the remaining participating companies in our study, 11 were SaaS providers, 5 provided infrastructure (IaaS) and 1 platform (PaaS), as shown in Table 2.1.

In this respect, we noticed a pattern in 66.7% of the IaaS and PaaS providers. Initially, they developed infrastructure and platform services to satisfy their internal company needs, and only later on exposed these on the market as cloud services. For example, one of the practitioners clarified the phenomenon: "we were normal [traditional] software providers and were paying for a virtual server

from [company name]; at some point, we decided to have our own [virtual server], then we developed in this direction even more and slowly became a IaaS provider ourselves".

Going into more details about this shift with our interviewees, we found that no real development strategy was implemented: requirements for the infrastructure service were collected ad-hoc internally and the service was provided for their own use. Later, when the service was sold outside the company, no further requirements elicitation activities were conducted, the service being sold as it was. Consequently, the organization started to provide a cloud service tailored to their own needs, but not necessarily to potential consumers' needs.

As far as SaaS providers are concerned, the situation is different. All of the 11 companies were traditional software providers at the time when they decided to become cloud providers. Therefore, they adapted their software such that it can be sold on demand, subscription-based, built the SLAs, and released the new cloud service. However, this shift happened only at the technical, implementation level, and virtually no attention was paid to the potential differences the cloud model may pose, in 63.6% of the cases. Therefore, no real requirements elicitation process took place. As new features or completely new cloud services were released, the elicitation methods of the past, the traditional software provider age, were used.

In summary, the dissatisfaction with the experienced elicitation alternatives is justified by the lack of evolution strategy, from traditional to cloud providers. On the one hand, known practices were used to acquire requirements only internally, within the company.

On the other hand, providers tried to use the techniques they were accustomed to also in the cloud, not paying attention to any adjustments or changes which may be needed.

2.5 Summary and Discussion

This section explains how the findings presented in Section 2.4 are inter-related and, based on these, draws the main conclusions for future research in requirements elicitation methods for cloud providers. All the key findings are summarized in Table 2.3.

The main goal of our study was to gain a deeper understanding on how cloud providers perform requirements elicitation, what the most popular techniques are, how these are implemented, what criteria and reasoning are used for choosing them, and the associated level of satisfaction.

We found that a significant number of the elicitation methods implemented by cloud providers are well-known approaches used by traditional software suppliers. For example, our study shows that traditional approaches and prototyping are the most frequently used elicitation techniques among cloud providers (A.1). In spite of being rated as popular, practitioners admitted that traditional methods, as well as most of the other existing techniques, are very difficult to virtually impossible to apply in the cloud context (A.2), leading to general disappointment and dissatisfaction (C.1). Their popularity is usually justified by the rigid internal processes and

Table 2.3: Key Findings.

A	**Requirements elicitation methods in use - RQ 1**
A.1	Traditional approaches (interviews, questionnaires, analysis of existing documentation, surveys) and prototyping are the most popular and highly applied existing requirements elicitation methods among cloud providers.
A.2	Most of the existing methods are very difficult to nearly impossible to apply in the cloud context.
A.3	Almost all cloud providers (94.7%) use ad-hoc elicitation methods at some point during their process. Moreover, the general criterion for method selection is also ad-hoc.
B	**Cloud versus traditional providers' methods - RQ 2**
B.1	The cloud calls for methods which fit for more heterogeneous audiences, take less time, and can be applied remotely.
B.2	Automation is needed to a larger extent for eliciting cloud consumer requirements.
C	**Do the existing methods suffice? - RQ 3**
C.1	Cloud providers' satisfaction level with regard to implementing existing elicitation approaches is low to medium.
C.2	In more than half of the cases, public cloud providers' dissatisfaction is caused by their evolution history.

regulations of the companies, and not by any suitability assessment or success stories. For example, if a provider used interviews in the past and it recently started to deliver cloud services along with its off-the-shelf software, it tries to apply the same methods also for the new scenario (C.2).

In most situations, these attempts fail due to new constraints posed by the cloud context, e.g., heterogeneous consumers located remotely. This often makes the synchronization impossible, thus leading to serious problems, such as failed projects or lost business opportunities. As explained in section 2.4 B, the elicitation methods for cloud providers should apply for diverse consumers, enable a shorter time to market, be applied remotely and ideally asynchronously (B.1), and make more use of automation (B.2).

Noticing these new needs should be met, cloud providers tried to adapt the existing techniques or even create new, ad-hoc approaches which can support them in eliciting requirements from the new type of consumers. Therefore, an impressive majority (94.7%) of our respondents reported to use ad-hoc methods during their elicitation processes (A.3), to compensate for the lack of existing suitable techniques. However, none of these ad-hoc methods stand out as very popular, as a sign that standardization is still far from being reality. This finding also confirms other researchers' findings: since the cloud paradigm has only been around for about five years [BYV+09], there are no standardized processes to govern how requirements engineering activities should be conducted by cloud providers. In particular, there are no dedicated elicitation methods tailored to the new cloud providers' needs [ZB11].

Based on the current state of practice described by our findings, we see a need to research new ways in which cloud providers can be supported in eliciting consumer requirements, taking into account the constraints and issues identified.

On the one hand, the development of elicitation methods should go towards adapting existing techniques, such as the traditional and market-driven, to suit the specific cloud properties (e.g. the distributed nature, on-demand delivery model), where possible. Ideally, cloud providers will restrict the use of currently available methods to only those which are well-suited and applicable in the cloud. This way, dissatisfaction will be avoided, as well as irrelevant output caused by method inadequacy. On the other hand, the current ad-hoc trials should evolve towards standardized methods, which can be used on a large scale.

Moreover, there is a need to investigate how the cloud paradigm enables cloud-specific requirements elicitation methods which go beyond what is known and used today. For example, measuring usage points on the service provided (B.2) can be considered an early step in this direction, but there is a wide range of other areas to explore.

2.6 Related Work

There is a large body of work on requirements elicitation; see [ZC05] for a comprehensive overview. Several studies point out the

strengths and limitations of requirements elicitation approaches in different settings [LRA02, Tuu03, KDNoD+03]. Other research addresses requirements elicitation in market-driven [KDNoD+03], [Saw00], distributed [DZ03] and asynchronous settings [GS05].

Our work goes beyond the state-of-the-art knowledge by revealing what kind of requirements elicitation techniques are used in the cloud context. Furthermore, we identify ad-hoc requirements elicitation approaches which are introduced by cloud providers in order to complement shortcomings of existing approaches. Moreover, our study confirms results of previous research in the field of market-driven RE [DKP+03, APC06, KDNoD+03]. For example, cloud service providers also use invented requirements, as mentioned by Potts [Pot95].

Our results are also in line with research on distributed RE. For example, Tuunanen [Tuu03] points out that reaching and involving stakeholders is a key problem in distributed contexts. He observes that involving so-called wide audience end-users, who are not within organizational reach, is not addressed by traditional requirements elicitation techniques. Other research explains that the identification of heterogeneous users is critical and needed [NE00], and that adequately identifying stakeholders in environments where the nature of stakeholders varies is challenging. Lim et al. [LQF10] present first ideas regarding this problem by highlighting an asynchronous and distributed stakeholder identification approach. In their work, they assume that key stakeholders are known and that they can identify other relevant stakeholders based on their domain knowledge. Our research demonstrates that this

challenge is also true for requirements elicitation in the cloud, and that sophisticated methods regarding the identification of heterogeneous stakeholders and their needs are needed.

Another key problem is that most well-known requirements elicitation methods support the elicitation from a predefined, limited number of stakeholders [Her07]. Particularly in the cloud context the number of stakeholders might be beyond of what traditional methods can support. Researchers highlight the need for user-driven approaches, which encourage users to actively push their needs to software developers, and therefore allow the involvement of large numbers of stakeholders in requirements elicitation [SGM10b]. However, current approaches focus on identifying ideas for new applications rather than communicating more detailed requirements [SGM10b].

2.7 Conclusion and Future Work

This paper reports on the results of an industrial exploratory study on requirements elicitation techniques used by cloud providers. The study involves 19 companies from 10 countries, represented by a total of 24 respondents. Our main contribution is to reveal which elicitation approaches are used in cloud systems, their limitations, and new challenges posed by the cloud paradigm with regard to requirements elicitation. Moreover, we identify key features of future dedicated methods that can address the issues observed.

Since the study was directly conducted with practitioners, we consider the results relevant for both industry and academia. For industry, they are an assessment of the state of practice from which cloud providers can understand the general adoption level of best practices and learn from the experiences of other organizations. For research, our work draws some possible directions for the future and shows new areas which need to be further explored and supported with new methods and tools.

In our future work, we will exploit these results to further investigate what elicitation methods would best support cloud providers in consumers' requirements acquisition.

Chapter 3

Scrutinizing Advanced Search Queries for Cloud Services with Fuzzy Galois Lattices

Original publication:
Quest for Requirements: Scrutinizing Advanced Search Queries for Cloud Services with Fuzzy Galois Lattices
Irina Todoran and Martin Glinz
10^{th} IEEE World Congress on Services (SERVICES'14)

Abstract

In software and requirements engineering, requirements elicitation is considered an essential step towards building successful systems. Despite extensive existing research in the field of distributed requirements engineering, the topic of requirements elicitation for

cloud systems *remains still uncovered. Cloud challenges (e.g., heterogeneous and globally distributed users, volatile requirements, frequent change requests) cannot always be satisfied by existing methods. We present a new approach for eliciting requirements for cloud services by analyzing advanced search queries. Our approach builds fuzzy Galois lattices for the terms that compose advanced search queries, thus enabling a thorough analysis of stored search data. This can support cloud providers in observing requirements clusters and new classes of cloud services, identifying the threshold for achieving satisfied consumers with a minimal set of requirements implemented, and thus designing novel solutions, based on market trends. Moreover, the Galois lattices approach enables large-scale consumers' involvement and ensures the elicitation of real requirements unobtrusively.*

3.1 Introduction

Requirements elicitation, that is seeking, capturing and consolidating requirements, is a core activity in any requirements engineering (RE) process [SS97]. Therefore, numerous elicitation techniques have been developed and are in use nowadays[ZC05]. Using elicitation techniques that do not fit the characteristics of the project increases RE costs and makes the project failure-prone [MR96, Tuu03]. Hence, approaches have been investigated for assigning techniques to contexts and selecting appropriate methods for individual cases [MR96]. Researchers have also provided comparisons [ZC05] and best practices on how to use these methods [SS97].

However, most existing techniques mainly address settings where stakeholders can be identified and analysts can directly interact with them. In today's context of *cloud systems*[BYV+09], traditional requirements elicitation techniques are heavily challenged [LNH07]. For instance, the communication with consumers becomes too expensive or even impossible because the key consumers are no longer known in person, being both too numerous and too heterogeneous. Since cloud consumers are often also globally distributed, it is virtually impossible to consider specific individual stakeholders.

Despite the rapid growth of the cloud use and the high number of cloud services available, dedicated requirements elicitation methods for the cloud are lacking [ZB11]. As a result, cloud service providers

have tried to adapt traditional methods such as workshops and interviews to work in distributed settings, e.g., by organizing online workshops and VoIP interviews supported by rich media [TSG13]. Others have used artificial stakeholders invented by marketing or substituted the global stakeholder community by a few pilot customers. Nevertheless, our previous research shows that such approaches have been rather unsuccessful so far [TSG13]. This finding is also supported by other researchers, who consider that the existing methods provide insufficient support or are difficult to apply in practice [YT03, LRA02]. Therefore, the cloud calls for thoroughly different requirements elicitation methods.

To address this research gap, we are investigating the possibility to infer new cloud service requirements from advanced search queries performed by (potential) cloud consumers. An *advanced search query* is a query that goes beyond simple keyword search by providing search information in some structured form. We exploit a particular form of advanced search queries where, for a given set of service features, users specify desired values for all those features (see Section 3.3.2 for an example). The data can be collected either on cloud providers' websites or on platforms that aggregate services from multiple cloud suppliers (marketplaces [ACL+12]), provided that they expose advanced search capabilities for the services available.

In this paper, we present our approach. We consider a given set S of search queries, each of them specifying desired values for a set F of service features. We first convert this information into a fuzzy binary relation $\tilde{R}(S, F)$. From this relation, we construct

a fuzzy Galois lattice [Bel99], which can then be analyzed with respect to three purposes: (i) understanding how requirements of (potential) service consumers can be clustered, (ii) identifying the threshold for achieving satisfied consumers with a minimal set of requirements implemented, and (iii) identifying new classes of cloud services, based on market trends. Our approach contributes a novel requirements elicitation technique which is specifically tailored to a cloud computing context.

The remainder of this paper is organized as follows. Section 3.2 summarizes the related work in the area of requirements elicitation techniques, potentially applicable in the cloud. Section 3.3 presents our new approach and applies it on a concrete example. The outcome is discussed in Section 3.4, and Section 3.5 concludes the paper.

3.2 Related Work

From the early 2000s, researchers observed that requirements engineering also needs to consider distributed [DZ03] and asynchronous settings [GS05], and this currently extends to the cloud context. However, due to its collaboration-intensive and time-consuming nature, requirements elicitation becomes difficult in the cloud [DZ03, Dau00].

As far as *dedicated cloud requirements elicitation methods* are concerned, there has been some advancement during the recent

years. For instance, frameworks focusing on the supply-demand relation have been designed [Liu12], and management systems for requirements ensuring QoS have been developed [VS11]. Moreover, researchers looked into methods for eliciting particular types of requirements, e.g., legal [BFS12] or security [BHC+13]. Still, these are only niche recommendations and no comprehensive clear solution exists, addressing the cloud-specific requirements elicitation challenges.

As far as *distributed requirements elicitation* is concerned, Lloyd et al. conducted a study [LRA02] on the effectiveness of elicitation techniques in distributed requirements engineering, concluding that synchronous elicitation approaches are generally more effective than asynchronous ones. Lim et al. [LQF10] present ideas on asynchronous and distributed stakeholder identification, assuming that key stakeholders are known, and further users can be identified based on domain knowledge. However, such approaches do not easily extend to the cloud context, since the audience for services is most often unknown and globally distributed.

Tuunanen [Tuu03] addresses the problem of reaching and involving wide audience end-users, or users who are not within organizational reach. He argues that traditional techniques do not provide adequate solutions and presents methods which could potentially fill this gap (e.g., EasyWinWin). However, none of these methods has been successfully used on a large scale for distributed elicitation so far. Moreover, research on EasyWinWin by Kukreja et al. [KB12] promises to provide support for distributed settings, but only focuses on stakeholders within organizational reach.

Another challenge of requirements elicitation in the cloud is the continuous change of consumer needs [Her07]. Consequently, various wiki approaches have been implemented, to provide a time-efficient possibility for updating and eliciting requirements. For instance, Decker et al. developed a wiki-based solution that enables stakeholders' participation in RE [DRR+07], Solis and Ali's spatial hypertext wiki focuses on distributed teams [SA10], whereas Liang et al. [LAC09] and Lohmann et al. [LRAZ08] exploit semantic annotation wikis. However, wiki-based methods generally assume that stakeholders are at least identifiable, which is not the case in a cloud context.

Studies from the field of web-information systems by Yang and Tang [YT03] reveal that the elicitation needs regarding Internet-based systems are also rather different from those of traditional systems (e.g., due to higher user diversity). Moreover, most existing requirements elicitation methods can only deal with a limited number of stakeholders [Her07]. The number of potential cloud service consumers may often go beyond what traditional methods can handle [BHC+13], and no real solutions addressing this elicitation issue have been developed so far. Market-driven techniques [KDNoD+03], which are usually employed when it is impossible to consider individual consumers, prove to be rather limited in the cloud, due to the lack of specific localized markets.

Work in data mining, machine learning and particularly recommender systems [AT05] also addresses the problem of extracting value from search data. For example, search-based and collaborative techniques can make personalized online product recommendations [AXG11], and user feedback has been used to rank

various products [LPP08]. Throughout the recent years, recommender systems [RV97] (e.g., probability-based collaborative filtering [HKTR04]) and clustering data mining methods [Ber06] have been heavily used for marketing purposes, to suggest similar products in e-commerce systems, or to segment populations. However, to our knowledge, such techniques have never been adapted or utilized for requirements elicitation in the cloud.

To summarize, the existing elicitation techniques, even when adapted, are mostly unsuitable for the cloud, and can support cloud service providers only to a limited extent in their requirements elicitation processes.

3.3 Our New Approach: Fuzzy Galois Lattice Analysis for Cloud Requirements Elicitation

Based on the existing related work and our previous research [TSG13], we found there is an evident need for dedicated requirements elicitation methods for the cloud. These should meet the following requirements:

R1: Fit for wider and heterogeneous audiences;

R2: Take less time than traditional elicitation methods;

R3: Make automated elicitation possible;

R4: Be applied remotely;

R5: Be able to handle volatile requirements.

To satisfy these requirements, we propose analyzing the data collected by cloud service providers or marketplaces in the form of advanced search queries, to infer new consumer requirements. For this, we represent the search queries by a fuzzy Galois lattice [Bel99]. Galois lattices allow the identification of sensible groupings of objects with common attributes. Due to these clustering and hierarchy properties [Wil09], Galois lattices have been used in software engineering for object identification [SMLD97], software modularization and analysis [SR99], and browsing software component libraries [Fis00]. However, to our knowledge, these properties have not been explored for new requirements acquisition based on advanced search queries.

3.3.1 Mathematical Foundations

In this section, we briefly review the mathematical foundations needed for our approach.

A partially ordered set (poset) A is called a *lattice* iff for any subset A' of A, there exist a least upper bound $sup \in A$ (the *supremum* of A') and a least lower bound $inf \in A$ (the *infimum* of A'). The supremum of A' is the smallest element of A that is greater than or equal to each element of A'. It is unique and it may or may not belong to A'. The infimum of A' is defined analogously. Lattices

can be graphically represented as acyclic directed graphs having exactly one source node (with no incoming edges) and one sink node (with no outgoing edges).

Galois connections have their roots in Galois theory [DEW13, EKMS93], and refer to correspondences between two partially ordered sets (posets). If $(A, \leq)$ and $(B, \leq)$ are two posets, a monotone Galois connection between them consists of two monotone functions $F : A \rightarrow B$ and $G : B \rightarrow A$, such that for all $a \in A$ and $b \in B$, we have $F(a) \leq b$ iff $a \leq G(b)$. The posets can be represented hierarchically in a graded system of sub- and superconcepts, which follows the mathematical axioms of a lattice. In a Galois concept lattice, the elements can generally take binary values. For a detailed overview on Galois theory, refer to [DEW13, EKMS93].

A *binary relation* $R(X, Y)$ is a set of ordered pairs $(x, y), x \in X, y \in Y$. For any given elements $p \in X$ and $q \in Y$, the pair (p, q) is either an element of $R(X, Y)$ or it is not. *Fuzzy binary relations* $\tilde{R}$ extend this digital behavior by allowing a *degree of membership* in a relation: the degree of membership of $(p, q) in \tilde{R}$ may be any real number from the interval $[0, 1]$. Obviously, the special case where every pair has a membership value of either zero or one represents a normal (crisp) binary relation.

In contrast to the general formal concept analysis theory (FCA) [GWF97], Galois connections take into consideration the relations between fuzzy concepts represented on ratio scales. For example, 0.2 and 0.5 are only two distinct values in FCA, whereas 0.2 and

0.5 are two values which can be ordered in Galois connections theory, e.g., 0.2<0.5. This leads to the notion of fuzzy Galois lattices [Bel99] the nodes of which represent fuzzy concepts, which in turn are constructed from a fuzzy binary relation.

3.3.2 A Running Example

We illustrate our approach with a concrete example of advanced searches for cloud data storage services, based on ten features. The first two columns of Table 3.2 list the features considered. While performing advanced search queries, consumers specify values for these features, based on their needs.

When we model cloud service queries, some features can be easily represented using only binary values, e.g., the service provides mobile support (1) or not (0). However, numerous features are better represented on ratio scales, e.g., for data storage cloud services, the values can be between 1 GB and 20 TB; in this case, binary values would be difficult to use. Therefore, we use the extension of the Galois lattice theory to fuzzy binary relations [Bel99], such that features can not only be represented on a nominal scale (taking the binary values 0 or 1), but also on a ratio scale. Therefore, a fuzzy set S_i includes a degree of membership for each of its elements, taking a value in the range [0,1]. A set with the membership degrees restricted to the values 0 and 1 (crisp set) is a particular type of a fuzzy set, so it is formally correct to mix the features on a nominal scale with those on a ratio scale in the same representation.

For example, the feature "f_1: Private user" is represented on a nominal scale (N) and can take the value 0, if the service is not available for private users, or 1 if the service is offered for private users. The feature "f_3: Storage" is represented on a ratio scale (R) and can take fuzzy values in the range [0,1].

3.3.3 The Steps of the Approach

In this section, we present the mechanics of our approach, consisting of seven steps. For each step, we first explain how it works in theory, and then apply it to our running example.

Conceptually, each advanced search query is composed of a set of cloud service features, which are specified by a (potential) consumer who is searching for a cloud service. In Galois theory, such a set is called a formal concept (FC). All search queries of a considered dataset are modelled as FCs in a Galois lattice, similar to the nodes of a graph. Moreover, the supremum and infimum elements of sub-lattices of the main lattice are calculated and also represented as FCs.

A fuzzy Galois lattice of a set of advanced search queries for cloud services can be generated and analyzed in seven steps, as follows. These are also summarized in Table 3.1.

Step 1. Having access to a search platform with advanced search capabilities and pre-defined possible features and values, data is collected from (potential) cloud consumers. Service features

Table 3.1: Fuzzy Galois lattice analysis for cloud requirements elicitation - Summary.

Step	Description of the step
1.	Collect advanced search queries for cloud services via a search platform.
2.	Model the search queries as formal concepts.
3.	Represent query data as a fuzzy binary relation.
4.	Calculate all the FCs for the given set of service queries.
5.	Analyze the fuzzy binary relation for special properties, apply reductions where possible.
6.	Represent the FCs in a fuzzy Galois concept lattice.
7.	Analyze the lattice nodes, the supremum and infimum elements for sub-lattices and potential clusters leading to new requirements for cloud services.

may include both functional (e.g., storage) and non-functional requirements (e.g., reliability).

Example: we collect advanced search queries for data storage services, using a marketplace for cloud services [Tod12]. This allows (potential) cloud service consumers to input their needs using predefined advanced search criteria.

Step 2. The search queries are modeled as fuzzy FCs using monotonic modeling functions.

Example: let S be a set (the universe of discourse) that denotes a generic data storage cloud service, with ten predefined features on the search platform, that users can opt for, as shown in Table 3.2. For each of these features, cloud providers can define monotonic modeling functions $f(x)$ that transform the numerical values input

Table 3.2: Service features (N = nominal scale: {0,1}, R = ratio scale: [0,1]).

No.	Feature	Scale	Modeling function f(x)
f_1	Private user	N	0: N/A, 1: available
f_2	Business user	N	0: N/A, 1: available
f_3	Storage	R	$10^{-3}x, x < 10^3\ GB$; $1, x \geq 10^3\ GB$
f_4	Mobile support	N	0: N/A, 1: available
f_5	File recovery	R	$10^{-2}x, x < 10^2 days$; $1, x \geq 10^2 days$
f_6	Reliability	R	$0, x \leq 90\%; 1, x > 99\%$; $0.1 * (x - 90), x \in (90, 99\%]$
f_7	AES encryption	N	0: N/A, 1: available
f_8	SSL encryption	N	0: N/A, 1: available
f_9	Max size/file	R	$0, x < 0.1\ GB; 1, x \geq 10\ GB$ $0.1 * x, x \in [0.1, 10)\ GB$
f_{10}	Uptime	R	$0, x \leq 90\%; 1, x > 99\%$; $0.1 * (x - 90), x \in (90, 99\%]$

by users into fuzzy values. For example, for the feature "storage capacity" of a cloud storage service, we can have a monotonic function as follows:

$$f(x) = \begin{cases} 10^{-3}x, & x < 10^3\ GB \\ 1, & x \geq 10^3\ GB \end{cases}$$

Accordingly, a value of 500 GB will be transformed into the fuzzy value 0.5. For features represented on a nominal scale, such as "AES encryption", $f(x)$ can take the value of 1 if the feature is available, else 0. The rest of features work similarly. Naturally,

there are numerous ways in which $f(x)$ can be defined; the choice only has to ensure that it is a monotonic transformation which leads to values in the range [0,1], and maintains a ratio scale for likely fuzzy values. Then, an advanced query for a cloud service can be defined as a fuzzy set $S_i = \{f_{ij} : j = 1, n\}$, where f represents the features of the cloud service, and n is the number of features defined in the search platform for the generic type of service S.

Step 3. The data is represented in a matrix, as a fuzzy binary relation $\tilde{R}(S, F)$.

Example: our data consists of five queries, represented as $\tilde{R}(S, F)$ in Table 3.3. The search queries are shown as rows in the table: S_i, i=1,5. For instance, $S_2 = \{f_1/0, f_2/1, f_3/0.2, f_4/1, f_5/0.9, f_6/0.8, f_7/1, f_8/0, f_9/0.5, f_{10}/1\}$ is an example of a fuzzy set representing an advanced query for a service, with the fuzzy values $\{0, 1, 0.2, 1, 0.9, 0.8, 1, 0, 0.5, 1\}$. In practice, this means that a (potential) consumer made an advanced search for a data storage service which is available only for business users and not for private users,

Table 3.3: The fuzzy binary relation $\tilde{R}(S, F)$.

$\tilde{\mathbf{R}}$	$\mathbf{f_1}$	$\mathbf{f_2}$	$\mathbf{f_3}$	$\mathbf{f_4}$	$\mathbf{f_5}$	$\mathbf{f_6}$	$\mathbf{f_7}$	$\mathbf{f_8}$	$\mathbf{f_9}$	$\mathbf{f_{10}}$
$\mathbf{S_1}$	1	1	0.7	0	0.3	0.4	1	0	0.3	1
$\mathbf{S_2}$	0	1	0.2	1	0.9	0.8	1	0	0.5	1
$\mathbf{S_3}$	0	1	0.1	0	0.5	0	1	0	0	0.9
$\mathbf{S_4}$	1	0	1	0	0.9	0.3	1	1	0.8	0.7
$\mathbf{S_5}$	1	0	0.5	0	0.3	0.7	0	0	0.6	1

can store up to 200 GB, provides mobile support, can recover files which are up to 90 days old, has a reliability of 98%, uses AES encryption but does not use SSL encryption, the maximum size per file is 5 GB and has an uptime higher than 99%.

Step 4. For all elements of the power set $\mathcal{P}(S)$ of the set S of search queries, we calculate the fuzzy concepts FC, yielding 2^n FCs, where n is the number of queries. According to Galois connections theory, the FC belonging to a subset S' is calculated by taking the minima of all feature values of the queries contained in S'.

Example: we compute the complete list[3] of $2^5 = 32$ fuzzy formal concepts FC. Due to space constraints, only a partial list is shown in Table 3.4 - for the complete list, please refer to the link. The indices indicate the queries that each FC is constructed of, e.g., $FC_{1,2,4}$ is a formal concept constructed of FC_1, FC_2 and FC_4, through intermediary formal concepts $FC_{1,2}$, $FC_{1,4}$ and $FC_{2,4}$.

Step 5. In case the context (matrix) exposes special properties, these are considered at this stage. As a typical example, assume we detect a small distance between two rows of the matrix. Since this method is based on computing minimum values, detecting a search query which is the minimum of another will lead to reduction opportunities in the final lattice, i.e. some nodes do not have to be represented due to redundancy. Another example is if we have duplicate entries in the list of computed FCs.

Example: analyzing the matrix $\tilde{R}$, we notice that the minimum of service queries S_2 and S_3 is S_3, at a small distance for some of the

[3]http://www.ifi.uzh.ch/rerg/people/todoran/FC_Complete_List.pdf

Table 3.4: Fuzzy concepts calculated from $\tilde{R}$ (partial list).

Label	**Fuzzy concept**
$FC_{\emptyset}$	$\{f_1/1, f_2/1, f_3/1, f_4/1, f_5/0.9, f_6/0.8, f_7/1, f_8/1, f_9/0.8, f_{10}/1\}$
...	
FC_4	$\{f_1/1, f_2/0, f_3/1, f_4/0, f_5/0.9, f_6/0.3, f_7/1, f_8/1, f_9/0.8, f_{10}/0.7\}$
FC_5	$\{f_1/1, f_2/0, f_3/0.5, f_4/0, f_5/0.3, f_6/0.7, f_7/0, f_8/0, f_9/0.6, f_{10}/1\}$
...	
$FC_{4,5}$	$\{f_1/1, f_2/0, f_3/0.5, f_4/0, f_5/0.3, f_6/0.3, f_7/0, f_8/0, f_9/0.6, f_{10}/0.7\}$
...	
$FC_{1,4,5}$	$\{f_1/1, f_2/0, f_3/0.5, f_4/0, f_5/0.3, f_6/0.3, f_7/0, f_8/0, f_9/0.3, f_{10}/0.7\}$
$FC_{2,4,5}$	$\{f_1/0, f_2/0, f_3/0.2, f_4/0, f_5/0.3, f_6/0.3, f_7/0, f_8/0, f_9/0.5, f_{10}/0.7\}$
...	
$FC_{1,2,4,5}$	$\{f_1/0, f_2/0, f_3/0.2, f_4/0, f_5/0.3, f_6/0.3, f_7/0, f_8/0, f_9/0.3, f_{10}/0.7\}$
...	
$FC_{1,2,3,4,5}$	$\{f_1/0, f_2/0, f_3/0.1, f_4/0, f_5/0.3, f_6/0, f_7/0, f_8/0, f_9/0, f_{10}/0.7\}$

features. This leads to opportunities for reduction, since some FCs will have identical values. Therefore, we will generate a sub-lattice corresponding to $\tilde{R}$, where each distinct node appears only once, as shown in the FCs list resulted afer eliminating duplicates[4], e.g., $FC_3 = FC_{2,3}$. Further, analyzing the complete list of FCs generated, several other duplicates can be identified. For example, $FC_{1,3} = FC_{1,2,3}$, since the minimum of S_2 and S_3 is S_3. Moreover, $FC_{3,4} = FC_{2,3,4}$, $FC_{3,5} = FC_{1,3,5} = FC_{2,3,5} = FC_{1,2,3,5}$, $FC_{1,3,4} = FC_{1,2,3,4}$ and $FC_{3,4,5} = FC_{1,3,4,5} = FC_{2,3,4,5} = FC_{1,2,3,4,5}$.

Step 6. We represent the fuzzy FCs in a concept lattice.

Example: the unique fuzzy formal concepts resulted in Step 5 are graphically represented as a fuzzy Galois sub-lattice of $\tilde{R}$, as shown in Figure 3.1.

Step 7. Finally, we analyze the lattice: if/how the FCs cluster, new feature combinations potentially leading to new services that could be developed and that do not exist at the moment, the supremum and infimum for sub-latticess leading to new ideas for cloud services, that satisfy significant populations.

Example: the output of our approach is analyzed in the following section.

[4]http://www.ifi.uzh.ch/rerg/people/todoran/Unique_FC_List.pdf

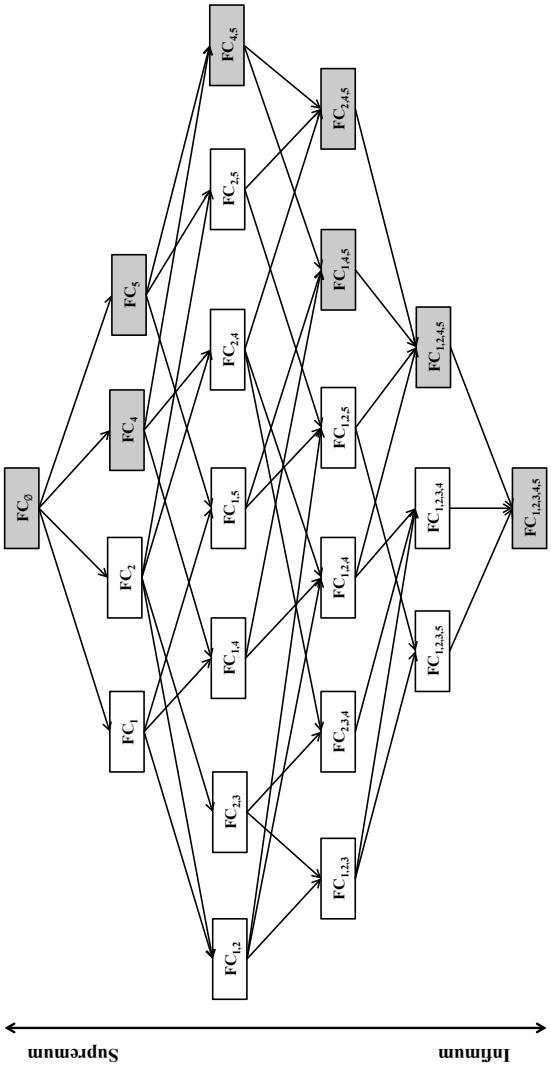

Figure 3.1: Fuzzy Galois sub-lattice of $\tilde{R}$. The grey FCs represent the sub-lattice SL discussed in Section 3.3.4.

3.3.4 Analysis of the Results

If a cloud provider wanted to fully satisfy only one query, i.e. satisfy all the features requested, exactly to the extents requested, it would have to supply a service having exactly the features specified in the query. However, this is unreasonable in most situations, since cloud providers cannot take into account individual wishes from each (potential) consumer, but rather target groups of consumers. Using our approach, for every subset of FCs, the supremum and infimum elements can be calculated. Practically speaking, the supremum of a set of FCs represents a comprehensive service that satisfies all the features requested in the corresponding subset of queries. For instance, if the cloud provider decides to consider all five queries, he could use the supremum of the queries, which is $FC_{\emptyset}$. Nevertheless, despite satisfying all queries, this may be impossible or too costly to implement. Therefore, we will analyze how the queries can be clustered and what minimum combinations of features would still achieve satisfied populations. For this, we analyze the infimum elements of sub-lattices. The infimum represents a cloud service which fully satisfies only those features that all queries in the corresponding subset have in common, while the rest are partially or not satisfied.

Since the space would not allow a complete analysis of the lattice generated in Step 6 (Figure 3.1), we choose to analyze a sub-lattice SL, e.g., for service queries S_4 and S_5. The relevant formal concepts for this sub-lattice are: the empty set $FC_{\emptyset}$, FC_4 and FC_5 which are a 1:1 mapping of the advanced service queries,

and the infimum nodes computed by our approach, which include both queries S_4 and S_5: $FC_{4,5}, FC_{1,4,5}, FC_{2,4,5}, FC_{1,2,4,5}$ and $FC_{1,2,3,4,5}$. These are colored in grey in Figure 3.1.

For queries S_4 and S_5, we calculate how many features are fully and partially satisfied by the infimum elements of SL. The graph in Figure 3.2 shows that $FC_{4,5}$ fully satisfies five out of ten features for S_4: $FC_{4,5}$ models a service that is available for private users and not for business users, does not provide mobile support, has 93% reliability and 97% uptime. S_5 is fully satisfied to a higher rate, with eight out of ten features: $f_1 - f_5$ and $f_7 - f_9$. If we cumulate the fully and partially satisfied features, we find that $FC_{4,5}$ satisfies

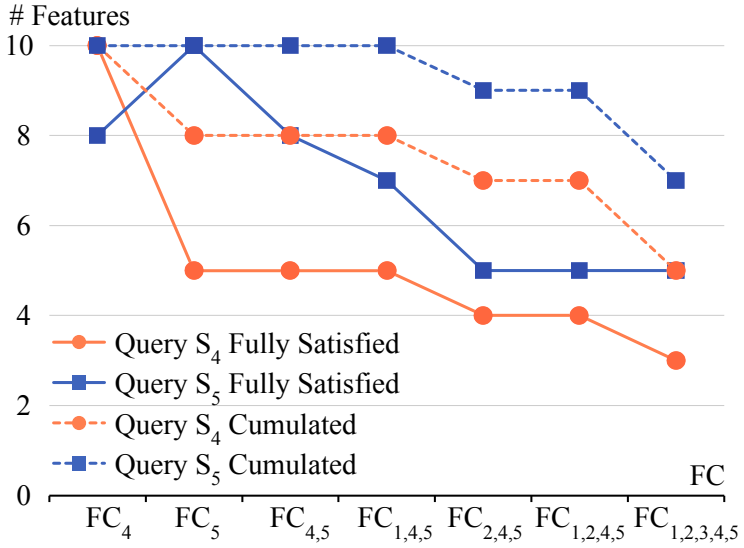

Figure 3.2: Number of features satisfied for queries S_4 and S_5.

eight features for S_4 and all features for S_5, respectively. Depending on the provider's strategy, this may already be a good compromise to address queries S_4 and S_5. Continuing the analysis, we observe that $FC_{1,4,5}$ does not differ significantly, since it fully satisfies the same features as $FC_{4,5}$ for S_4, and only changes the extent to which f_9 is satisfied for S_5: instead of allowing a maximum file size of 6 GB, $FC_{1,4,5}$ only allows files of maximum 3 GB. As far as the cumulated satisfaction is concerned, this is identical to the one reached with $FC_{4,5}$. Moreover, since $FC_{1,4,5}$ is also an infimum for FC_1, it will satisfy some of the features mentioned in S_1: five fully and three partially. Therefore, given the overall performance, it seems $FC_{1,4,5}$ would be a better choice than $FC_{4,5}$, if the cloud provider decided to supply a new service addressing service queries S_4 and S_5. Similarly, the fitness rates for $FC_{2,4,5}$, $FC_{1,2,4,5}$ and $FC_{1,2,3,4,5}$ are analyzed. The graph shows that $FC_{2,4,5}$ is less suitable than $FC_{1,4,5}$, when aiming to satisfy the authors of queries S_4 and S_5, with an advantage of only four fully and three partially satisfied features for S_2. $FC_{1,2,4,5}$ is equal in performance to $FC_{2,4,5}$ as far as S_4 and S_5 are concerned, but with a higher advantage: it satisfies four features fully and three partially for S_1, and also satisfies three features fully and four partially for S_2. Eventually, the number of features satisfied converges to zero, reaching zero if the input queries represent disjoint sets.

This analysis shows that S_4 and S_5 can be satisfied simultaneously by several combinations of features, which are more economical to implement than services addressing individual needs, still maintaining high satisfaction rates. Infimum elements representing

new classes of services such as $FC_{4,5}$, $FC_{1,4,5}$ and even $FC_{1,2,4,5}$ are possible compromises for achieving satisfied consumers with a minimal set of features implemented, depending on how thoroughly the cloud provider wants to consider the initial queries. Moreover, we noticed that S_1 clusters with S_4 and S_5 better than S_2, for example. It should be noted that the more we advance to the right in the graph in Figure 3.2, the lower the satisfaction level for the initial two queries, but the higher the advantages for other queries, since the FCs are infima of more queries. This is a demonstration example, but when larger amounts of data are analyzed and the complete lattice is considered, the results are naturally even more conclusive.

3.4 Discussion

The qualitative results of our approach are promising. Starting the analysis from two advanced search queries for cloud data storage services, our approach was able to identify a potential cluster of queries (e.g., S_1, S_4 and S_5), thus supporting cloud providers in understanding how their potential consumers can be grouped (i). Moreover, new classes of services emerged from the lattice analysis (iii), such as $FC_{1,4,5}$ and $FC_{1,2,4,5}$, showing possible thresholds for achieving satisfied consumers with a minimal set of requirements implemented (ii).

In Section 3.1, we introduced five requirements that should be met by dedicated requirements elicitation methods for the cloud. We now evaluate how our approach satisfies them.

R1: fit for wider and heterogeneous audiences. The advanced searches needed by our fuzzy Galois lattices technique are always conducted on cloud providers' or marketplaces' websites. Therefore, our approach allows any number of (potential) consumers from virtually anywhere to input their needs for services. This is done in a completely asynchronous way, without the need of a requirements engineer supervising the requirements elicitation process.

R2: take less time than traditional elicitation methods. The approach introduced has a passive character, i.e. consumers are not directly and consciously involved in the requirements elicitation process, since the requirements for new services are inferred based on their searches. This way, virtually no time is dedicated specifically to the elicitation process, but rather to the data analysis.

R3: make automated elicitation possible. Our technique is tool-supported, such that most of the analysis is automated. Steps 1-6 are purely automated, and Step 7 is semi-automated. Whereas the new classes of services are automatically generated, while performing the analysis in Step 7, providers can perform a manual what-if analysis to dynamically simulate what happens when only one or a few features are varied, how these impact the general clustering, or zoom in specific parts of the lattice, to analyze the best ideas for new services.

R4: be applied remotely. Since this is a search-based approach, it can be applied for any consumers, located anywhere, including those who are not physically reachable. Given its unobtrusive character, the technique is also suitable when consumers would

not be able to describe their requirements easily in an interview or a workshop.

R5: be able to handle volatile requirements. Our approach enables a continuous elicitation process, since data is collected permanently from consumers who perform advanced searches. This feature is useful for monitoring volatile requirements, which makes the technique especially fitting with the agile character of most cloud provider companies.

There is no perfect, general-purpose requirements elicitation method - each has its strengths and weaknesses and performs best in a particular context or domain [NE00]. Our approach is best-suited for the early elicitation phase, and for monitoring market trends. It can be succeeded by more in-depth requirements elicitation with complementary methods such as prototyping and large-scale online experiments. Moreover, having generated the Galois lattice for a set of queries, this can be used by cloud service providers to evaluate where their existing offering is positioned in the spectrum of requested similar services on the market. This can then be used to compute how the existing offering can be enhanced, for example, to fit the needs of a larger population.

A limitation of our approach is that it assumes consumers provide values for all the features specified as advanced search criteria. For example, if a user specifies values of zero or no values for all features, this leads to infima equal to zero, meaning services with no features. This problem can be solved by ignoring the queries having values of zero for all features (outliers) and by allocating default values for all the features with no values assigned.

As far as scalability is concerned, the tool which is curently under development needs three seconds to compute the Galois lattice for 200 advanced search queries for cloud services with ten features, on a 4 GB 1333 MHz DDR3, 1.8 GHz Intel Core i7. A more in-depth analysis of scalability is subject to future work.

3.5 Conclusion and Future Work

This work presents an approach for infering new cloud service requirements based on what consumers look for, i.e. their advanced search queries. The approach produces fuzzy Galois lattices, composed of the initial consumer queries and their computed supremum and infimum elements. These lattices can help cloud service providers to analyze how the queries can be grouped to satisfy large populations with a minimum of implemented requirements, and to identify new classes of services needed on the market.

We plan to enhance our approach by releasing a tool that supports the technique introduced, such that cloud providers can benefit from the automatic features. Moreover, we plan to develop a concrete formalism for calculating the satisfaction level for certain combinations of requirements, and to evaluate the approach with real-world data, in order to detect potential shortcomings when working with large datasets from cloud providers.

Chapter 4

StakeCloud Tool: From Cloud Consumers' Search Queries to New Service Requirements

Original publication:
StakeCloud Tool: From Cloud Consumers' Search Queries to New Service Requirements
Irina Todoran Koitz and Martin Glinz
23rd IEEE International Requirements Engineering Conference (RE'15)

Abstract

Requirements elicitation is indispensable for delivering successful services. Nevertheless, cloud service providers mostly rely on ad-hoc approaches, as there are no dedicated elicitation methods for cloud services. To address this problem, we developed the StakeCloud approach, which helps cloud providers elicit requirements

for future cloud services. StakeCloud builds and analyzes fuzzy Galois lattices based on consumers' advanced search queries for cloud services. Our StakeCloud Tool automatically builds the lattice from the given search queries. It provides the requirements analyst with extensive clustering and analysis capabilities as well as means for comparing different newly generated classes of services. These allow identifying the threshold for achieving the largest populations of satisfied consumers with a minimum set of features implemented. Further, our tool enables eliciting real requirements from global consumers unobtrusively.

Related video available at: https://goo.gl/PcNLPw

4.1 Introduction

Understanding consumers' requirements is critical for building successful services [SS97]. Therefore, a wide range of requirements elicitation methods has been developed and used in practice over the last decades. However, in the domain of cloud services, the existing techniques are challenged [TSG13]. The cloud is not only bringing technical and economic benefits [LKN+09], but also difficulties regarding requirements elicitation, e.g., heterogeneous and globally distributed consumers. According to our previous work [TSG13], the most popular requirements elicitation techniques among cloud providers are traditional methods (questionnaires, surveys, interviews, document analysis) and prototyping. Nevertheless, these are inadequate for the cloud: e.g., they require long elicitation times, can rarely be applied remotely and do not support automation. Hence cloud service providers use ad-hoc approaches such as inventing, guessing requirements or imitating competitors, which lead to dissatisfaction or failure-prone services.

To address this issue, we introduced the StakeCloud approach for eliciting cloud service requirements [TG14]. With the emergence of marketplaces as search platforms for cloud services (e.g., Intel Cloud Finder), cloud consumers have the opportunity to input their advanced search queries for cloud services to find solutions that match their needs. Based on such queries represented by finite sets of feature values collected on marketplaces, StakeCloud models fuzzy Galois lattices which are then analyzed to infer new cloud service requirements. Lattices are represented graphically

as acyclic directed graphs having exactly one source node with no incoming edges and one sink node with no outgoing edges (for an example, see Figure 4.2). In lattices, any two nodes have at least one supremum and one infimum element. For more information on our theoretical approach, please refer to [TG14]. In this tool demo paper, we describe the StakeCloud Tool and how it can be used by cloud providers in their decision-making process. Our main contribution is a novel tool-supported requirements elicitation technique tailored to a cloud computing context.

4.2 The StakeCloud Tool

Starting from our theoretical solution [TG14], we developed the StakeCloud Tool, which automates the requirements elicitation activity for cloud services to a large extent, as shown in Figure 4.1. Once the queries are collected (A1), the tool checks for duplicates (A2) and counts the individual frequencies (A3). Then, the cloud provider analyst models the queries as fuzzy vectors (A4: this activity is partially automated by the tool) and decides whether (s)he wants to cluster similar queries (A5). If applicable, the tool computes the clusters (A6) and then generates the Galois lattice automatically (A7). At this stage, the cloud provider can iteratively choose from and apply the available analysis criteria (A8). The tool will automatically generate the corresponding graphs (A9), indicating what combinations of features and feature values satisfy the largest number of input consumer queries with a minimum compromise. These represent draft requirements for new services

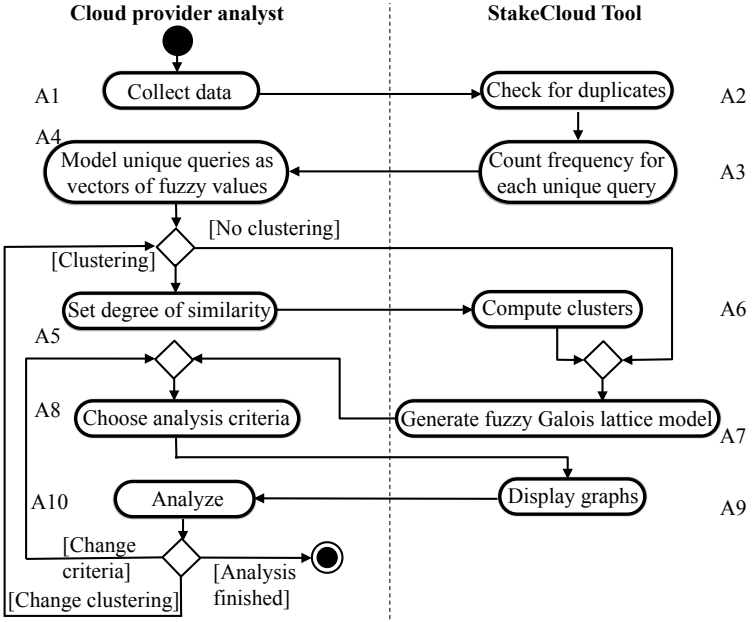

Figure 4.1: UML activity diagram of the process of our approach.

that the cloud provider representative can use in his/her decision-making process regarding upcoming service releases. Naturally, our approach and tool can be complemented by existing more in-depth requirements elicitation techniques.

Figure 4.2 presents a screenshot of the StakeCloud Tool, where the main aspects are indicated with green numbers (1-5). Once an input file containing consumers' advanced search queries (encoded as fuzzy vectors) is loaded, the corresponding fuzzy Galois lattice is *automatically generated* and *displayed* in the main panel of the window (1). The elements in the first level of the lattice hierarchy, i.e., nodes identified by only one digit, represent the initial search queries, whereas the other elements represent new classes of potential cloud services generated by our approach. The user (cloud provider) can *select* (e.g., nodes *[4]* and *[5]*) and *drag* nodes, choose to *display the actual fuzzy values* composing the selected nodes or *highlight the suprema and infima elements* of particular nodes by clicking the buttons in the lower part of the window (2). The supremum is a comprehensive service that satisfies all the features requested in the corresponding subset of queries. However, this is most often impossible or too expensive to implement. Hence the infimum elements are better candidates for development, since they fully satisfy only those features that all queries in the corresponding subset have in common, while the remaining ones are partially or not satisfied. Our tool provides compromise analysis support regarding the implementation of requirements for future services.

If the cloud provider analyst is interested in seeing where his/her offering is situated in the lattice among the queries and generated

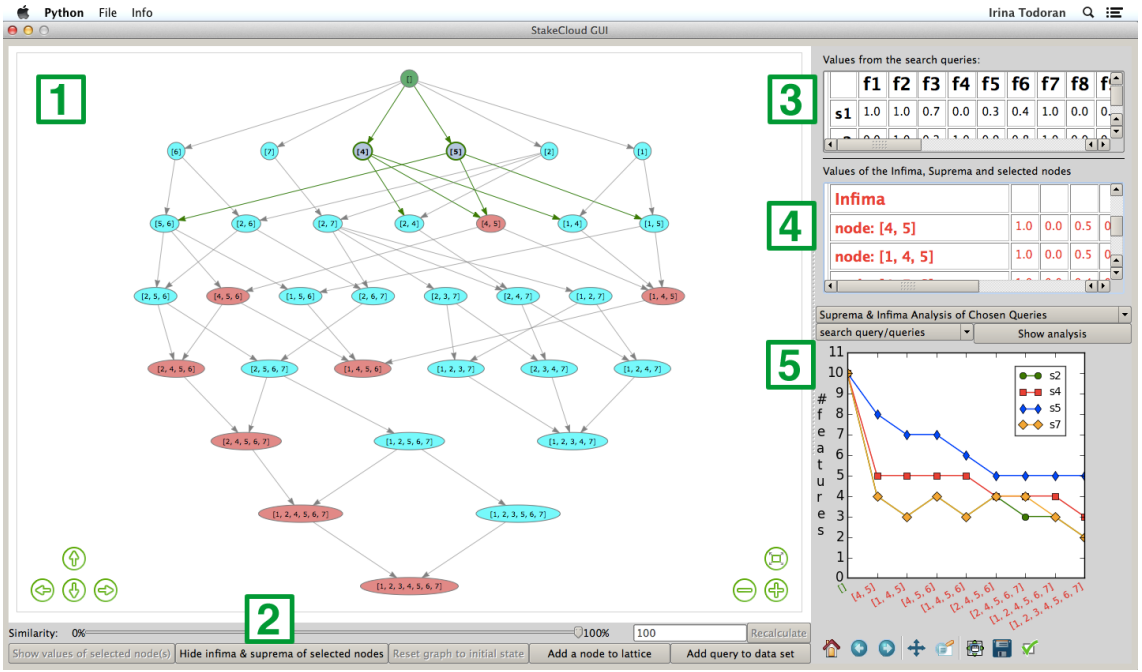

Figure 4.2: StakeCloud Tool screenshot.

classes of services, (s)he can choose to *add a node to the lattice* (2) by typing the feature values for his/her service. Moreover, in case the lattice is too large for a good visualization, the *zooming* functionalities or the *similarity degree* can be used. By setting a similarity degree (2), similar queries are *clustered*, thus reducing the overall lattice size. However, the composing queries can still be analyzed and viewed in detail. In the right side of the window, the top panel (3) *shows the values of the input search queries*, and the middle panel (4) displays the fuzzy vectors for *selected nodes* or *infima* (red) and *suprema nodes* (green). The bottom panel (5) exhibits *five lattice analysis options* in a drop-down menu, e.g., analyze to what extent individual features are accomplished by the selected nodes, or how many features are fully satisfied by their suprema/infima. The video at http://goo.gl/qv5I5s shows a practical scenario of the StakeCloud Tool.

As far as the technical implementation is concerned, our tool was developed in Python 2.7, using PyQt4 for the graphical user interface. Since PyQT is a Qt binding for Python, the StakeCloud tool is cross-platform and has been tested in MacOS 10.x, Windows 8 and Linux Ubuntu 14.10. For visualizing the lattice, the vis.js library was used, whereas the analysis graphs are matplotlib graphs.

4.3 Conclusion and Future Work

The StakeCloud Tool implements the fuzzy Galois lattices approach for requirements elicitation in cloud settings. It allows

analyzing advanced search queries for cloud services to infer new requirements, thus semi-automating the elicitation activity. This way, our approach and tool enable large-scale consumers' involvement and is best suited for the early elicitation phase and for monitoring market trends. So far, the technical performance of the StakeCloud tool has been evaluated with self-generated datasets, yielding promising results, and we plan to further use the tool with cloud provider companies.

Chapter 5

Evaluating the Fuzzy Galois Lattices Approach to Requirements Elicitation for Cloud Services

Original publication:
A Fuzzy Galois Lattices Approach to Requirements Elicitation for Cloud Services
Irina Todoran Koitz and Martin Glinz
IEEE Transactions on Services Computing (TSC), 2015

Abstract

The cloud paradigm has become increasingly attractive throughout the recent years due to its both technical and economic foreseen impact. Therefore, researchers and practitioners attention has

been drawn to enhancing the technological characteristics of cloud services, such as performance, scalability or security. However, the topic of identifying and understanding cloud consumers real needs has largely been ignored. Existing requirements elicitation methods are not appropriate for the cloud computing domain, where consumers are highly heterogeneous and geographically distributed, have frequent change requests and expect services to be delivered at a fast pace. In this paper, we introduce a new approach to requirements elicitation for cloud services, which utilizes consumers advanced search queries for services to infer requirements that can lead to new cloud solutions. For this, starting from the queries, we build fuzzy Galois lattices that can be used by public cloud providers to analyze market needs and trends, as well as optimum solutions for satisfying the largest populations possible with a minimum set of features implemented. This new approach complements the existing requirements elicitation techniques in that it is a dedicated cloud method which operates with data that already exists, without entailing the active participation of consumers and requirements specialists.

5.1 Introduction

Cloud computing is largely seen as a successful and promising paradigm due to its capability to efficiently adapt to business changes by scaling software or hardware resources in a flexible way. Therefore, it has received great interest from both research and industry throughout the recent years, and has so far managed to maintain its promise to deliver both technical and economic benefits [LKN+09].

As a result, the number of public cloud services available is growing continuously and it is expected that this growth will continue in the future [CLPZ11]. While this can be seen as an advantage for cloud consumers who have a wide variety of offers to choose from, this phenomenon can also lead to a paradox of choice [Sch04], where users do not know what services best match their needs. To solve this problem, researchers [KS10a] and industry practitioners recognize that there is a need to develop search engines or platforms dedicated to aggregating and displaying cloud service offerings from various providers. These would act as marketplaces [AT05, Tod12] exposing advanced search capabilities that allow (potential) cloud consumers to input and refine their needs according to various criteria. Then, a matching algorithm would identify what existing public cloud services match the features requested. An example in this direction is the Intel Cloud Finder [Int15], which matches IT requirements to existing cloud services from those providers that signed up on the Intel platform and published their offering.

Another consequence of the rapid growth in popularity of cloud services is the emergence of numerous related research areas, ranging from intercloud architecture models to cloud performance and virtualization. In this context, the focus has been strongly directed towards building better services from a technological perspective, whereas the human aspect has been largely ignored. While there are examples of research conducted in the area of cloud adoption[KHGSS12, ZBE14], the issue of identifying and satisfying cloud consumers' real needs has not been thoroughly addressed. Moreover, the existing requirements acquisition or elicitation techniques are not suitable for the cloud domain, and dedicated requirements elicitation methods for the cloud are lacking [ZB11]. In this paper, we introduce a new approach for solving an existing requirements engineering (RE) problem: the absence of dedicated requirements elicitation techniques for cloud services. We concentrate on identifying (potential) cloud consumers' needs by modeling and analyzing the data collected from consumers' advanced search queries on cloud service marketplaces.

In the following sections, we first explain what requirements elicitation is and why it is important. Then, we clarify the cloud challenges that hinder the usage of existing requirements elicitation methods and finally outline the main contributions of this paper.

5.1.1 Requirements Elicitation

Requirements elicitation is typically seen as the first step in the requirements engineering process [NE00]. According to Sommerville

and Kotonya, requirements elicitation refers to activities undertaken to discover the requirements of a system to be built or a problem to be solved [SK98]. Additionally, van Lamsweerde also includes the identification of stakeholders in the requirements elicitation stage [vL09]. Generally, requirements elicitation refers to seeking, gathering and consolidating requirements, and is regarded as an indispensable step towards building successful solutions.

Nuseibeh et al. highlight that requirements are not somewhere, waiting to be collected, but elicitation techniques are necessary to investigate and understand users' needs [NER00]. For instance, traditional methods (e.g., questionnaires, interviews, analysis of existing documentation), group elicitation methods (e.g., brainstorming, focus groups, RAD/JAD workshops), prototyping, model-driven techniques (e.g., scenarios, KAOS, i*), cognitive methods (e.g., protocol analysis, laddering, card sorting) and contextual techniques were developed to enable requirements elicitation, and have been used successfully in traditional settings for decades.

Similarly, in the cloud domain, consumers' requirements have to be identified in order to know what characteristics future cloud services should exhibit such that they satisfy consumers' needs, and to avoid failure-proneness. However, the cloud paradigm poses a few challenges that do not allow using the existing requirements elicitation methods, as explained in the following section.

5.1.2 Cloud Challenges

Whereas existing requirements elicitation methods have proven useful for determining stakeholders' needs in traditional contexts, i.e. where stakeholders are easy to identify and physically reachable, most of these techniques are heavily challenged by the particular features of the cloud. For instance, given that cloud services can be easily sold and customized online, consumers are generally *geographically distributed*, often worldwide. This leads to a *lack of local markets*, i.e. cloud providers do not always have a deep understanding of the international markets they sell to. This is in contrast to the traditional delivery model, where contracts are made with local physical suppliers that then sell software or hardware solutions, facilitating this way the expansion of the business to a local, known market.

Moreover, cloud consumers can be highly *numerous and heterogeneous*, with diverse profiles and backgrounds, and exhibiting thoroughly different requirements that cannot be easily satisfied on an individual basis. In such settings, existing elicitation methods cannot be applied, since stakeholders cannot be identified, such that requirements specialists can interact with them directly [LRA02].

Another challenge is represented by the *frequent change requests* coming from cloud consumers, especially businesses, and their *volatile requirements*. On the one hand, such cloud consumers are largely modern businesses that need their requests to be met fast. On the other hand, to face the competition, providers have to first

know, ideally predict, and then satisfy these requests efficiently, such that they do not lose their clients. In this context, they cannot afford to apply elicitation methods that require long waiting times for gathering requirements, or long processing and analyzing times, and this is where most of the existing techniques fail.

Last but not least, the cloud is still *young* compared to the traditional delivery model. Therefore, dedicated methods for addressing consumers' needs have not been developed so far, to address the challenges identified.

Tsumaki and Tamai [TT06] categorize the existing requirements elicitation methods according to two criteria. Firstly, depending on how requirements acquisition is conducted, requirements can be collected and sorted either in a static or dynamic way. Secondly, depending on the properties of the target space to be analyzed, the space can be either closed or open. Using this categorization, due to the fast and dynamic pace of the cloud, service providers should elicit requirements in a *dynamic*, ideally continuous fashion. Since consumers' needs may change rapidly and this can often be unpredictable, the space is *open*. As it can be observed in Figure 5.1, methods such as brainstorming, role playing or ethnography could seemingly fit these characteristics. However, these are the type of methods that necessarily require the physical and simultaneous presence of stakeholders in the same geographical space, which is incompatible with the cloud paradigm. Therefore, according to the existing related work and based on our previous research [TSG13], it is evident that there is a need for dedicated cloud elicitation techniques that support cloud companies in understanding and

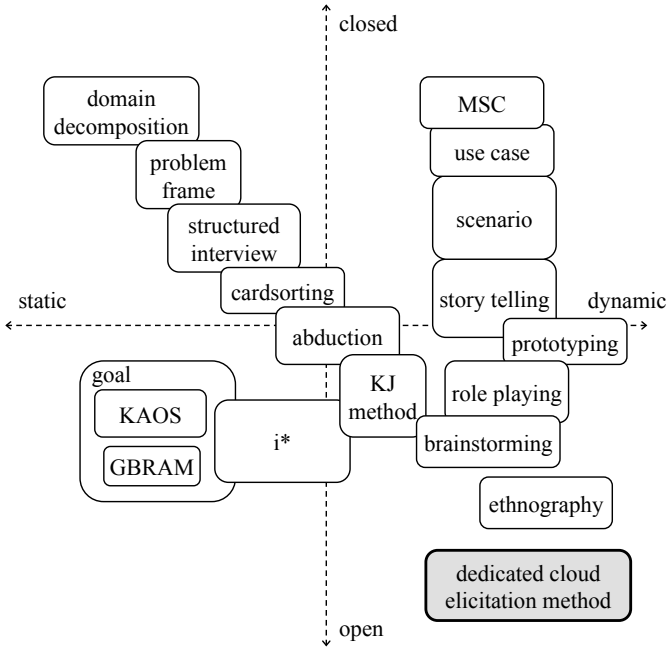

Figure 5.1: Requirements elicitation techniques, adapted after [TT06].

providing their consumers with services that truly meet their needs. Such a dedicated method would belong in the framework proposed by Tsumaki and Tamai in the bottom right corner (dynamic elicitation process and open space), as shown in Figure 5.1, and should accommodate the following requirements [TSG13]:

R1: Fit for wide and heterogeneous audiences;

R2: Take less time than traditional elicitation methods;

R3: Make automated elicitation possible;

R4: Be applied remotely;

R5: Be able to handle volatile requirements.

5.1.3 Main Contributions

This work is an extension of an existing paper published in the 2014 IEEE 10th World Congress on Services [TG14]. There, we introduced the preliminary idea of building fuzzy Galois lattices to support cloud providers in the requirements elicitation activity. In this extension, we have three new contributions:

1. We enhance the preliminary algorithm introduced in [TG14], by improving its performance through a pre-processing technique that calculates frequencies of search queries. This is included as a functionality in the StakeCloud Tool we developed.

2. We add a similarity classifier that allows the flexible clustering of similar queries; this leads to reducing the modeling space, thus improving the scalability. Similarly, this functionality is included in the prototype.

3. We conduct a series of experimental evaluations to verify our requirements elicitation approach, and explain how it meets the five requirements above.

The remainder of the paper is organized as follows. Section 5.2 describes our approach, including the idea, definitions for the terms used and algorithms developed. Section 5.3 presents the evaluation of the solution and the interpretation of the results. In Section 5.4, we give an overview of related work, and Section 5.5 concludes the paper.

5.2 Approach

5.2.1 Definitions

In this section, we introduce the terms utilized in describing our approach, as well as the corresponding mathematical definitions.

Definition 1. *A* partially ordered set (or poset) *is a set taken together with a partial order on it. Formally, a partially ordered set is defined as an ordered pair $A = (X, \leq)$, where X is called the ground set of A and $\leq$ is the partial order of A.*

Definition 2. *For any subset A' of a poset A, the members of the families $lb(A') = \{a \in A : \forall a' \in A' : a \leq a'\}$ and $ub(A') = \{a \in A : \forall a' \in A' : a' \leq a\}$ are called the* lower *and* upper bounds *of A' in A, respectively.*

Definition 3. *The members of the families $inf(A') = max(lb(A'))$ and $sup(A') = min(ub(A'))$ are called* infima *and* suprema *of A' in A, respectively.*

In other words, the *supremum* of A' is the smallest element of A that is greater than or equal to each element of A'. It is unique and it may or may not belong to A'. The *infimum* of A' is defined analogously.

Definition 4. *A poset A is called a* Galois lattice *iff for any subset A' of A, there exist a least upper bound $sup \in A$ (the* supremum *of A') and a least lower bound $inf \in A$ (the* infimum *of A').*

Galois connections have their roots in Galois theory [DEW13], and refer to correspondences between two posets. According to Erné et al.[EKMS93], they are defined as follows:

Definition 5. *Considering the posets $\mathcal{P} = \langle P, \leq \rangle$ and $\mathcal{D} = \langle Q, \geq \rangle$, if $P \overset{\pi_*}{\to} Q$ and $Q \overset{\pi^*}{\to} P$ are functions such that for all $p \in P$ and all $q \in Q$, $p \leq q\pi^*$ iff $p\pi^* \sqsubseteq q$, then the quadruple $\pi = \langle \mathcal{P}, \pi_*, \pi^*, \mathcal{D} \rangle$ is called a* Galois connection.

Definition 6. *A* binary relation *$R(X, Y)$ is a set of ordered pairs $(x, y), x \in X, y \in Y$. For any given elements $p \in X$ and $q \in Y$, the pair (p, q) is either an element of $R(X, Y)$ or it is not.*

In a Galois concept lattice, the elements can generally take binary values. When we model cloud service queries, some features can be easily represented using only binary values, e.g., the service provides mobile support (1) or not (0). However, numerous features are better represented on ratio scales, e.g., for data storage cloud services, the values can be between 1 GB and 20 TB; in this case, binary values and therefore binary relations would be difficult to

use. Consequently, we use the extension of the Galois lattice theory to fuzzy binary relations [Bel99], such that features can not only be represented on a nominal scale (taking the binary values 0 or 1), but also on a ratio scale.

Definition 7. Fuzzy binary relations $\tilde{R}$ *allow a* degree of membership *in a relation: the degree of membership of* (p, q) *in* $\tilde{R}$ *may be any real number from the range* $[0, 1]$.

Therefore, a fuzzy set S_i includes a degree of membership for each of its elements, taking a value in the range [0,1]. A set with the membership degrees restricted to the values 0 and 1 (crisp set) is a particular type of a fuzzy set, so it is formally correct to mix the features on a nominal scale with those on a ratio scale in the same representation.

Definition 8. *Each concept in a Galois hierarchy that represents the set of objects sharing the same values for a certain set of properties is called a* formal concept.

In contrast to the general formal concept analysis theory (FCA) [GWF97], Galois connections take into consideration the relations between fuzzy concepts represented on ratio scales. For example, 0.2 and 0.5 are only two distinct values in FCA, whereas 0.2 and 0.5 are two values which can be ordered in Galois connections theory, e.g., 0.2<0.5. This leads to the notion of fuzzy Galois lattices [Bel99] the nodes of which represent fuzzy concepts, which in turn are constructed from a fuzzy binary relation.

5.2.2 Idea

With the emergence of cloud service marketplaces acting as intermediators between consumers and providers, large amounts of data are generated. (Potential) consumers use such platforms to search for services that match their criteria, thus inputting their needs and preferences as advanced search queries. Such log data are currently used by recommender systems [RV97] to suggest existing services with similar features, but this is generally the highest extent to which these data are exploited. A significant majority of the companies we interviewed in one of our previous studies [TSG13] mentioned that they logged consumers' search data and tried to analyze it, but the large dimensions usually hindered the understanding. In most cases, their analysis was reduced to identifying which service features appeared most commonly in the advanced search queries. Therefore, we are facing a big data problem: the large datasets cannot be processed and visualized easily, which leads to losing the potential of such data.

Our idea is to utilize the logged advanced search queries for cloud services to find requirements and combinations of features that can eventually lead to developing new cloud services and new classes of cloud solutions. Our approach follows the process illustrated by the UML activity diagram in Figure 5.2. The process contains ten activities, as follows.

A1: Collect data. The input necessary for running our approach is represented by advanced search queries data collected through a cloud services marketplace. In this context, an advanced search

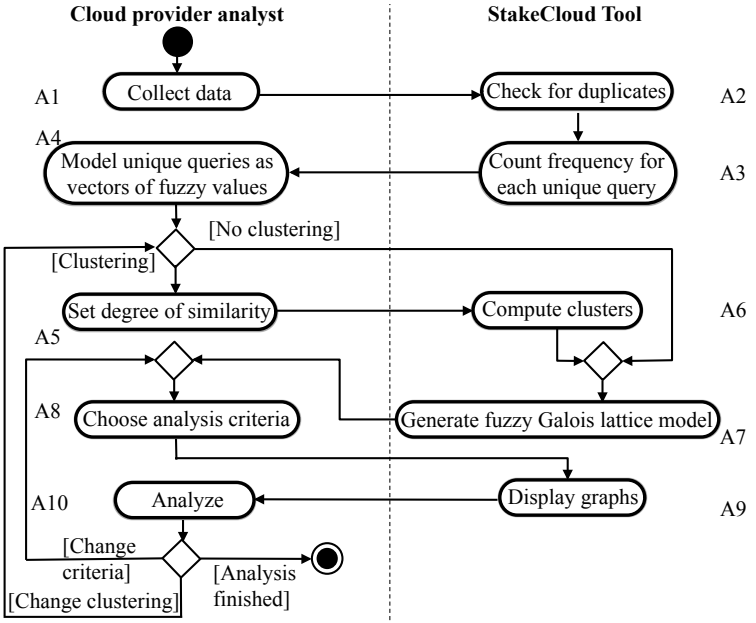

Figure 5.2: UML activity diagram of the process of our approach [TKG15b].

query is a query that allows users to specify desired values for a set of predefined features.

A2: Check for duplicates. Once the data has been collected, our StakeCloud Tool that implements the approach checks the input dataset for duplicates, i.e. identical queries.

A3: Count frequency for each unique query. In case no duplicate queries are found, each query has the frequency set to one. If duplicate queries are identified, the tool keeps only one instance of

each unique query and counts its frequency, i.e. how many times it appears in the given dataset.

A4: Model unique queries as vectors of fuzzy values. Naturally, advanced search queries contain heterogeneous data. For instance, the values associated with a query input for a public cloud data storage service can be: 300GB storage capacity, 30 days file recovery, 98 percent reliability, and the service should allow AES encryption for backup. Using monotonic modeling functions, such values are transformed into fuzzy values, taking values in the range [0,1]. For example, the feature "reliability" can be modeled as a monotonic function as follows, transforming a value of 95 percent into the fuzzy value 0.5. Here, the range (90,99 percent] is considered the most important and is modeled accordingly, since most services do not have reliability values under 90 percent or above 99 percent.

$$f(x) = \begin{cases} 0, x \leq 90\% \\ 0.1 * (x - 90), x \in (90, 99\%] \\ 1, x > 99\% \end{cases}$$

For features represented on a nominal scale, such as "AES encryption", $f(x)$ can take the value of 1 if the feature is available, else 0. The rest of features work similarly. Evidently, there are numerous ways in which $f(x)$ can be defined; the choice only has to ensure that it is a monotonic transformation which leads to values in the range [0,1], and maintains a ratio scale for fuzzy values. Table 5.1 shows a list of ten different features of a cloud data storage

Table 5.1: Service features (N = nominal scale: {0,1}, R = ratio scale: [0,1]).

No.	Feature	Scale	Modeling function f(x)
f_1	Private user	N	0: N/A; 1: available
f_2	Business user	N	0: N/A; 1: available
f_3	Storage	R	$10^{-3}x, x < 10^3\ GB$; $1, x \geq 10^3\ GB$
f_4	Mobile support	N	0: N/A; 1: available
f_5	File recovery	R	$10^{-2}x, x < 10^2 days$; $1, x \geq 10^2 days$
f_6	Reliability	R	$0, x \leq 90\%; 1, x > 99\%$; $0.1 * (x - 90), x \in (90, 99\%]$
f_7	AES encryption	N	0: N/A; 1: available
f_8	SSL encryption	N	0: N/A; 1: available
f_9	Max size/file	R	$0, x < 0.1\ GB; 1, x \geq 10\ GB$; $0.1 * x, x \in [0.1, 10)\ GB$
f_{10}	Uptime	R	$0, x \leq 90\%; 1, x > 99\%$; $0.1 * (x - 90), x \in (90, 99\%]$

service, along with possible modeling functions. This way, an advanced query for a cloud service can be defined as a vector of fuzzy values $S_i = \{f_{ij} : j = 1, n\}$, where f represents the features of the cloud service, and n is the number of features assigned to the generic type of service S in the search platform. According to Figure 5.2, the activity of defining the modeling functions is performed by the cloud provider. However, our tool is enhanced with a set of pre-defined monotonic functions for common cloud service features, which can be used by the cloud provider to (semi-) automate this task. Moreover, the tool has a checking mechanism that ensures the functions defined by the provider cover the range

[0,1] monotonously. The current tool does not support modeling exclusion queries, but we plan to integrate this in our next release. The undesired features can be marked in our dataset with a flag and then propagated in the lattices.

A5: Set degree of similarity. The cloud provider representative can bundle or cluster the input queries by selecting the degree of similarity on a scale from 0 to 100 percent. The degree of similarity is an integer in the range [0,100] and a similarity of 100 percent means that all the initial queries included in a cluster are 100 percent similar, i.e. identical. Therefore, when a dataset includes only unique advanced search queries, the clusters computed for a similarity degree of 100 percent coincide with the initial queries. A similarity degree of 0 percent means the queries included in the clusters are not similar at all, thus leading to one single cluster that includes all the queries in the dataset. In general, we define the similarity degree as a measure of the degree to which queries in a dataset cluster together: a value of n percent means that all the initial queries included in a cluster are n percent similar.

Provided that the degree of similarity is smaller than 100, the bundling action will group similar queries and define a representative vector of fuzzy values for each such bundle (more details on this in Section 5.2.3). The bundling is calculated based on Euclidean distances between the vectors of features composing the input queries and the supremum of all the queries. When the input datasets are large, this activity may be necessary to enable an easier visualization of the output and achieve improved performance. For this reason, we allow the user to set the degree

of similarity already before generating the lattice. However, the user can also directly generate the lattice without bundling.

A6: Compute bundles. Provided that the user selected a degree of similarity different from the default value (100 percent), the tool computes the corresponding bundles and the representatives (as vectors of fuzzy values) for each. The algorithm used for this step is detailed in Section 5.2.3.

A7: Generate fuzzy Galois lattice model. Based on the input dataset modeled as vectors of fuzzy values, and/or the computed bundles, the tool generates the corresponding fuzzy Galois lattice (for an example, see Figure 5.7). Lattices are represented graphically as acyclic directed graphs having exactly one source node (with no incoming edges) and one sink node (with no outgoing edges). In our applied case of fuzzy Galois lattices, the nodes on the first level in the hierarchy correspond to the vectors of features given by cloud consumers (e.g., *(1), (2)*). As explained in Section 5.2.1, the topmost element is the supremum or upper bound for all the lattice elements. This is the only lattice node that satisfies all input queries fully. However, implementing such a service in practice is most often either very expensive or impossible. Therefore, we recommend that cloud providers should analyze the infima options, which are represented by all the other nodes of the lattice, from the second hierarchy level down (e.g., *(1,4), (2,7), (2,4,7), (1,2,3,4)*). Infima nodes are service offerings that satisfy the input queries, but only to a limited extent. This way, providers can make compromises to satisfy large populations of consumers with a minimum set of requirements implemented, to reach an

optimum solution. In this respect, it is important to note that all the infima elements represent newly generated classes of services, which can be candidates for implementation. Therefore, our approach goes beyond simple statistics, since the infima elements are new combinations of feature values, which cannot be easily inferred by counting frequencies in the initial datasets or using standard statistical methods. Moreover, if the lattice generated is split in sub-lattices for a more thorough analysis, the same properties related to infima and suprema elements are maintained for the sub-models: any sub-lattice has at least a supremum and an infimum node, respectively, so in any such subset of nodes there will be at least a service candidate that fully satisfies the nodes in the first hierarchy level and at least a node that satisfies these to a limited extent, which can be calculated by our method. The algorithm employed for building these models is described in Section 5.2.3.

A8: Choose analysis criteria. Once the lattice has been drawn (including bundles or not), the cloud provider user of our approach can choose the preferred criteria for performing data analysis. These criteria are selected and implemented in such a way that they support the service provider in his/her understanding and reasoning process, to decide what types of services should be supplied to satisfy consumers' needs. For instance, (s)he can choose to compare the satisfaction level for individual features for a set of classes of services (graph nodes) (s)he selected in the lattice, or (s)he can study to what extent particular queries can be satisfied by new classes of services generated by our method. These analysis criteria are detailed in Section 5.3.5.

A9: Display graphs. Based on the criteria chosen, the corresponding graphs are displayed. The graphical representation can consist of points or functions in a Cartesian coordinate system (e.g., as shown in Figure 5.7).

A10: Analyze. The cloud company representative can use the lattice model and graphical representations from A9 to perform a thorough analysis of consumers' queries. Moreover, (s)he can change the analysis criteria at any time and generate new graphs, or change the degree of similarity used for bundling.

5.2.3 Algorithms

The activities that compose the process of our approach introduced in Section 5.2.2 are implemented by the following algorithms.

Frequency Counter

In order to support activities A2 and A3, we utilize Algorithm 1, shown in pseudocode below. This calculates the frequency for each query of the input file.

As an abstract data type, we use a dictionary (or associative array) composed of $(key, value)$ pairs. In our case, the values contained by the dictionary are vectors of fuzzy values ($fuzzyVector$), and each key appears only once. For every query q of the initial input

Algorithm 1 calculateFrequency(inputFile)

```
1: d = dictionary(fuzzyVector)
2: foreach query q ∈ inputFile do
3:     if q ∉ d.keys() then
4:         d[q] = 1
5:     else
6:         d[q] = d[q] + 1
7:     end if
8: end for
```

file, we check whether it is part of the keys set or not. If it is found, we increment its frequency counter; if not, the frequency is set to one. The output consists of a dataset made of exclusively unique queries and the corresponding computed frequencies. This algorithm ensures that the dataset to be processed further does not contain any duplicates, which is important for the overall performance and behaviour of the method. Algorithm 1 has $O(S)$ complexity, where S is the number of service queries.

Lattice Generator

The activity A7 is one of the core steps in our approach, since it deals with generating the fuzzy Galois lattice, the pre-requisite model for the data analysis. Algorithm 2 shows what operations are needed before the graph can be drawn.

The input of Algorithm 2 is represented by a fuzzy binary relation (fBR), which is a matrix with two dimensions: S service

Algorithm 2 generateLattice(fBR[S,F])

1: $C = C' = \emptyset$
2: $C_S^k = \binom{S}{k}$
3: $C = \cup_{k=1}^{S}\{C_S^k\}$
4: **foreach** $i \in C$ **do**
5: **foreach** $j \in i$ **do**
6: **foreach** $f \in [1..F]$ **do**
7: $infFeature[f] = min_{r=1}^{len(j)} j[r, f]$
8: $C' = C' \cup infFeature[f]$
9: **end for**
10: **end for**
11: **end for**
12: eliminateDuplicates(C')
13: drawGraph(C')

queries that exist in the dataset with unique queries, and F features, which are the pre-defined features for the type of service analyzed, e.g., cloud data storage, as shown in Table 5.1. Firstly, all elements of the power set $\mathcal{P}(S)$ of the set S of search queries are generated as combinations (line 2). These are sets of sets of vectors with fuzzy values, where S queries are taken k at a time without repetition. For example, if $S = 5$ and $k = 3$, $C_5^3 = \{[s_1, s_2, s_3], [s_1, s_2, s_4], [s_1, s_2, s_5], [s_1, s_3, s_4], [s_1, s_3, s_5],\ [s_1, s_4, s_5], [s_2, s_3, s_4], [s_2, s_3, s_5], [s_2, s_4, s_5], [s_3, s_4, s_5]\}$, where $s_i, i = 1, S$ are unique service queries. Secondly, we append all these C_S^k calculated to C (line 3), which becomes a large set that includes all the possible combinations of service queries, based on the initial input file.

Thirdly, we calculate the fuzzy concepts (lines 4-11), yielding 2^S FCs. According to Galois connections theory, the FC belonging to

a subset S' is calculated by taking the minima of all feature values of the queries contained in S' (line 7). In case C exposes special properties, these are considered at this stage. As a typical example, assume we detect a small distance between two rows of the fuzzy binary relation. Since this method is based on computing minimum values, detecting a search query which is the minimum of another will lead to reduction opportunities in the final lattice, i.e. some nodes do not have to be represented due to redundancy.

Naturally, after calculating all the FCs, C' may include duplicates. In order not to draw any classes of services more than once, we now eliminate the duplicates from C'. This way, we keep only one entry for each fuzzy vector generated when calculating the feature values (line 7).

The last step of Algorithm 2 consists of drawing the lattice (graph). When doing so, the algorithm takes into account the hierarchical properties of the lattice. For instance, our approach will draw the directed edges $FC_4 - FC_{4,5}$ and $FC_5 - FC_{4,5}$ from service queries s_4 and s_5 represented by FC_4 and FC_5, respectively, to the lattice node $FC_{4,5}$, which is a formal concept generated based on queries s_4 and s_5. The other edges are drawn in a similar fashion, e.g., $FC_{4,5} - FC_{2,4,5}$, $FC_{2,4,5} - FC_{1,2,4,5}$, such that the indices of the supremum node always represent a subset of the indices of the infimum node.

The operation for calculating the minima values in line 7 has a complexity of $O(\frac{2^S}{S})$ which, combined with the complexity of iterating over the combinations $O(2^S - 1)$ and features $O(F)$, leads

to the overall complexity for Algorithm 2 of $O(\frac{F*2^{2S}}{S})$, therefore $O(2^S)$. The exponential character in the size of S is natural when computing Galois lattices.

Similarity Classifier

The third algorithm is used for implementing activity A6 (cf. Figure 5.2), where the tool computes clusters of similar queries based on a degree of similarity *sim* provided by the user in activity A5.

Algorithm 3 takes the degree of similarity *sim* and the set of unique queries (*uniqueQ*) as input. The latter is the output of activity A4, and is a bi-dimensional matrix with S queries and F service features. The algorithm returns the *clustersList*, which is a list containing all the clusters computed. Initially, this is initialized to the empty set. Our bundling algorithm uses Euclidean distances between vectors to find clusters, in a similar way to other clustering algorithms from data mining, such as k-means. Nevertheless, we could not have used an existing known implementation such as Lloyd's algorithm, since we cannot provide the number of expected clusters (the k value in k-means), but are interested in seeing the resulting clusters, regardless of their number. In this respect, our algorithm belongs to hierarchical clustering, avoiding the problem of first identifying the number of clusters generated. Moreover, it has a deterministic behavior, always producing the same results for the same input. Therefore, instead of selecting some random

Algorithm 3 calculateSimilarity(sim,uniqueQ[S,F])

1: $clustersList = \emptyset$
2: **if** $sim == 100$ **then**
3: return $uniqueQ[S, F]$
4: **else**
5: $sup = supremum(uniqueQ)$
6: $uniqueQ'[S', F'] = uniqueQ[S, F]$
7: **foreach** *query* $q \in uniqueQ$ **do**
8: $dist(sup, q) = \sqrt{\sum_{i=1}^{F}(sup_i - q_i)^2}$
9: **end for**
10: **while** $uniqueQ' \neq \emptyset$ **do**
11: $maxDist = \max_{i=1,S',q_i \in uniqueQ'}(dist(sup, q_i))$
12: $cluster = \emptyset$
13: $breakingPoint = \frac{sim * maxDist}{100}$
14: **foreach** *query* $q \in uniqueQ'$ **do**
15: **if** $dist(sup, q) > breakingPoint$ **then**
16: $cluster = cluster \cup q$
17: **end if**
18: **end for**
19: $uniqueQ' = uniqueQ' - \{q | q \in cluster\}$
20: $clustersList = clustersList \cup cluster$
21: **end while**
22: return $clustersList$
23: **end if**

queries as anchors for the clusters, we use one single anchor: the supremum of the input dataset *sup*.

In case the degree of similarity *sim* given by the user is equal to 100, the algorithm returns the list of initial unique queries (lines 2-3) - the clusters list for $sim == 100$ coincides with the initial dataset when all the input queries are unique. Otherwise, if the *sim* value is smaller than 100, we calculate the maximum Euclidean distance between the supremum *sup* and each query q in our set (lines 7-8), and the breaking point for the cluster (lines 11-13). The *breakingPoint* defines the lower bound of the cluster, i.e. the least Euclidean distance to the supremum within which a search query must be in order to qualify for cluster membership. All the queries whose distances to the supremum are greater than the *breakingPoint* value for a given *sim* are considered similar for the *sim* value specified. Each element of the set ends up in a cluster, as long as the distance between it and the supremum is greater than the breaking point (lines 14-18). Then, the dataset is updated (line 19) to include only those elements that did not become members of clusters or were left out as single unclustered elements, and the steps above are repeated until the dataset is empty. Finally, the clusters formed in each iteration are added to the *clustersList* (line 20).

For calculating the representative of each cluster (a vector of fuzzy values), we use the centroid or geometric center concept. This is calculated as the arithmetic mean position of all the points in the n-dimensional space, where n is the number of features for the specified cloud service. We did not choose the medoid concept as

the representative object for each cluster, as it is most commonly done in data mining, since medoids are always elements of the dataset, and this was not a requirement in our case. Here, it is more important to achieve a minimal dissimilarity between the representative and all the elements of the cluster, such that the new lattice generated after the bundling activity illustrates the initial dataset well, without high information loss. If the queries included in the cluster have frequencies higher than one, these are also taken into account, since we calculate a weighted centroid, i.e. each query is represented in the cluster proportionally to its frequency.

Given that we first test whether *sim* is 0, the best case complexity for Algorithm 3 is $O(1)$. Otherwise, since calculating $maxDist$ is done in $O(N')$, the overall complexity is $O(N + N'^2)$, where N is the number of elements of $uniqueQ$ and N' is the number of elements of $uniqueQ'$. The worst case complexity tends to $O(N^2)$, when N' tends to N.

5.3 Evaluation

5.3.1 Goal and Metrics

To evaluate our approach, we assess how it meets the five requirements introduced in Section 5.1.2 (R1-R5). These emerged from the related work in the field of dedicated cloud elicitation

techniques and our previous study with 19 cloud provider companies [TSG13]. Therefore, we assume that having an approach that meets these requirements would support cloud providers in understanding and satisfying their consumers' needs better. We define the goal for our evaluation as follows:

> Analyze *our dedicated requirements elicitation approach* for the purpose of *evaluation* with respect to *the extent to which it satisfies cloud providers' requirements for a new requirements elicitation method (R1-R5)* from the point of view of *cloud provider companies* in the context of *analyzing advanced search queries for cloud services to infer new requirements.*

Since R2 deals with time-efficiency, the main metric we use for meeting this requirement is the time. We calculate the time needed for generating the lattice nodes and models, as well as the time required for finding similar queries and generating clusters. As far as the automation is concerned (R3), we show what output can be automatically generated by our approach and explain to what extent the method introduced is more automatic than the existing requirements elicitation techniques. Furthermore, we use the standard deviation as a metric for heterogeneity (R1) of queries in a dataset. As far as remote application (R4) and volatile requirements (R5) are concerned, we describe how our approach can be applied and its output can be analyzed remotely, as well as how volatile requirements can be monitored and future predictions can be made.

5.3.2 Method

In this work, we focus on internal, rather than external evaluation. This means that we evaluate the performance of our approach and present how it meets the requirements identified (R1-R5), but do not conduct an external evaluation against other existing methods. The main reason for this is that such an evaluation is virtually impossible.

None of the existing requirements elicitation techniques mentioned in Section 5.1.1 uses advanced search queries datasets as input, such that we could compare the output of our method to the output of other similar methods. In this respect, the innovative nature of our approach is the cause for the lack of a benchmark or ground truth we could use for an external evaluation, e.g., using metrics such as precision, recall or fallout. Moreover, trying to analyze the datasets manually to build our own ground truth is impossible, given the large amounts of data.

5.3.3 Product Managers' Input

In our semi-structured interviews conducted with cloud providers between November 2012 and January 2013 on how they perform requirements elicitation [TSG13], we discussed the possibility of utilizing consumers' search data for infering service requirements. At that time, while some companies were logging such data, none of the 19 interviewed cloud businesses was using them for requirements elicitation. In January 2015, we contacted again five of

Table 5.2: Product Managers' ideas on using the sample dataset.

Activity	PM1	PM2	PM3	PM4	PM5
Analyze query frequency	✗	✗	✗		✗
Analyze query importance					✗
Predict future requests		✗		✗	
Analyze queries for services similar to theirs			✗	✗	

the product managers (PM) we had previously interviewed, from companies located in three different European countries, and asked them if this situation had changed meanwhile. All five explained that although the idea of using log data analysis for the purpose of getting to know their (potential) consumers better is a recurrent topic, no concrete initiatives have been taken in this regard so far. They confirmed that the requirements elicitation approaches they were using at the time we performed our interviews are still in use now.

Moreover, we now gave the five product managers a sample dataset of 500 advanced search queries, and asked them how they would use it to help their companies in their decision making process regarding the launch of a new cloud service. The most frequent responses recorded are displayed in Table 5.2. Analyzing the frequency of each query was the the most common activity they would undertake (4 out of 5 responses), reasoning that if numerous people search

for the same type of service, that is a sign such a service should be launched, if it does not exist. PM2 and PM4 mentioned that several such datasets from different moments in time could be used to predict future requests, based on the evolution history. PM3 and PM4 also added that it would be interesting to analyze those queries that are similar to what their companies already offer, to identify what kind of changes could be implemented to enhance their services and satisfy consumers. The most advanced approach was suggested by PM5, who mentioned she would use the tf-idf (term frequency-inverse document frequency) method to determine how important a query is in a set of queries (document) of a collection of sets of queries (corpus). This is similar to analyzing individual query frequency, since the tf-idf is proportional to the frequency of each query, but it is offset by the frequency of the query in the corpus. When asked whether they utilized such datasets at all, three PMs answered that their recommender systems are the only ones making use of such data, to suggest services similar to the ones viewed by the user.

Therefore, although the five PMs interviewed are too few for achieving statistical significance, they confirmed our hypothesis that there are no elicitation methods in use that take the same type of input as our method, for the purpose of requirements acquisition. Moreover, our tool implementing the approach supports automating the activities suggested by the PMs, and goes beyond these. On the one hand, these findings encourage us to continue our efforts invested in elaborating the presented approach. On the other hand, they certify that finding a ground truth for an external

evaluation is virtually impossible. Consequently, we performed the evaluation described below.

5.3.4 Experimental Setup

We used three distinct datasets as our simulation data, representing advanced search queries for data storage cloud services. They contain 250 queries[5], 500 queries[6] and 1000 queries[7], respectively. Since obtaining large amounts of real-world search information is particularly difficult due to sensitivity and privacy, we generated the needed datasets ourselves.

However, we used an available small dataset containing real-world data [TG14] coming from one of the companies interviewed [TSG13] as a starting point. Moreover, we generated the queries such that they define data storage cloud services with ten features, as described in Table 5.1. For this, we followed the constraints imposed by the individual features: some are represented on a nominal scale, whereas others are represented on a ratio scale. Therefore, some of the features are defined by binary values, such as the "mobile support", and others are defined by fuzzy values in the range [0,1], such as "reliability". To test the performance of the similarity classifier, we set a degree of similarity between 10 and 80 percent.

All the experiments presented in the following sections were run on a 4 GB 1333 MHz DDR3, 1.8 GHz Intel Core i7.

[5] http://www.ifi.uzh.ch/rerg/people/todoran/Dataset250.pdf
[6] http://www.ifi.uzh.ch/rerg/people/todoran/Dataset500.pdf
[7] http://www.ifi.uzh.ch/rerg/people/todoran/Dataset1000.pdf

5.3.5 Automation and Time-Efficiency

We saw that existing requirements elicitation methods are challenged in the cloud domain by the time factor. Moreover, the need to involve a large number of stakeholders simultaneously in the same geographical spot is another issue posed by the cloud settings, which could be solved by means of automation. To show how our approach addresses these challenges, we firstly analyze the time efficiency (R2) of our algorithms and then describe the automation capabilities (R3).

Time Efficiency (R2)

In contrast to the existing elicitation techniques, our approach has a passive character, i.e. consumers are not directly and consciously involved in the requirements elicitation process, since the requirements for new services are inferred based on their searches. This way, virtually no time is dedicated specifically to the elicitation process, but rather to the data analysis. Consequently, we will calculate the data processing times.

Similarity Classifier

We firstly analyze the behavior of the similarity classifier (Algorithm 3). As described in Section 5.2.3, the degree of similarity (sim) is given by the cloud provider representative using the StakeCloud prototype [TKG15b] that implements our approach. Then, the bundles of similar queries are automatically generated. We

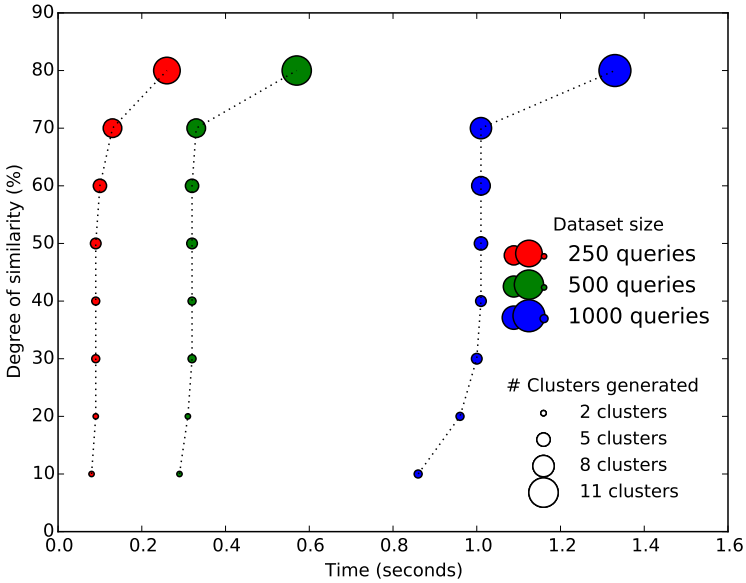

Figure 5.3: Time performance of the similarity classifier and number of clusters generated.

measured the time needed to generate clusters for degrees of similarity between 10 and 80 percent, with an increment of 10, for the three datasets of 250, 500 and 1000 queries, respectively. We excluded the extreme margins of the range [0,100], i.e. [0,10) and (80,100], because values converging to 100 will not generate any clusters and will simply maintain the original list of queries, and values converging to 0 will likely generate only one large cluster consisting of all the queries in the dataset. Moreover, we measured how many clusters are generated for each degree of similarity, for every dataset. The results obtained are shown in Figure 5.3.

Generally, the time needed to process the data and generate queries bundles grows linearly with the number of queries in the dataset: e.g., for $sim = 80$, generating clusters for a dataset of 250 queries (shown in red in Figure 5.3) will take 0.26 seconds, while for a dataset of 500 queries (green) it will take 0.57 seconds, and for 1000 queries (blue) 1.33 seconds. This observation also holds for the other values of the degree of similarity. The time values we obtained are in the same range as those of well-known clustering methods, such as k-means [CND11], for datasets of comparable sizes.

As expected, the time required to generate the clusters is directly proportional to the number of clusters generated and to the value of sim. In Figure 5.3, the bigger the size of the circle representing the cluster, the larger the number of clusters generated for that particular degree of similarity, as exemplified in the legend. For instance, for the dataset with 500 queries (green), $sim = 10$ will lead to 2 clusters, whereas $sim = 80$ will lead to eleven clusters. It can be noted that the value of $sim = 80$ causes comparatively longer needed time periods compared to the other values. This is due to the fact that our algorithm always compares the distances between the supremum and each query to the breaking point calculated (line 15, in Algorithm 3), and the higher the value of sim, the more such computations needed. Nevertheless, the times are still manageable on a regular machine, being in the range of seconds for thousands of queries.

According to our tests so far, visualizing Galois lattices generated directly from datasets with less than 15 queries is still possible,

without any need for clustering, unless specifically desired so. The tool allows zooming and dragging, thus making it easy for the user to navigate in the models even when the graphs are rather complex, as in Figure 5.4 b) and c). For datasets with more than 15 queries, the user can choose to automatically compute bundles and then generate the lattice. This way, (s)he can adjust the degree of similarity until the model is easy to visualize. When some of the represented nodes are not original queries from the dataset, but bundles representatives, a mouseover feature allows the user to see what composing queries each cluster consists of, and their frequencies.

Lattice Generation

We computed the time needed to generate fuzzy Galois lattices for various datasets, containing between 5 and 19 queries. We tested our Algorithm 2 with datasets of these sizes, since these are likely to lead to models that are still easy to visualize and analyze, as explained above. For this reason, we generally recommend bundling similar queries initially, if the input dataset contains more than 15 queries, such that an overall view of the data is first generated. Then, the user can explore individual bundles and regenerate the model as needed. This also ensures a rapid lattice generation, thus fast data processing: e.g., building the model for ten distinct queries takes 0.05 seconds, generating a total of 154 nodes to be represented in the lattice.

The graph in Figure 5.5 shows the time needed to generate the corresponding lattice nodes for the datasets used. The labels displayed next to the points represent the number of queries processed.

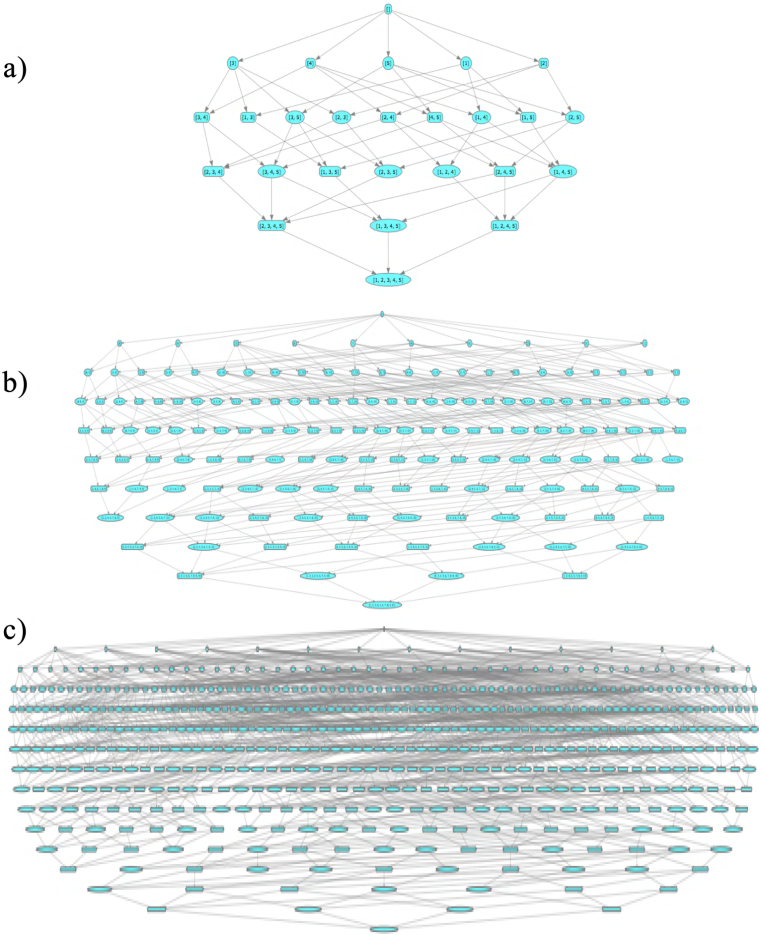

Figure 5.4: Fuzzy Galois lattices for: a) 5 queries, b) 10 queries, and c) 15 queries.

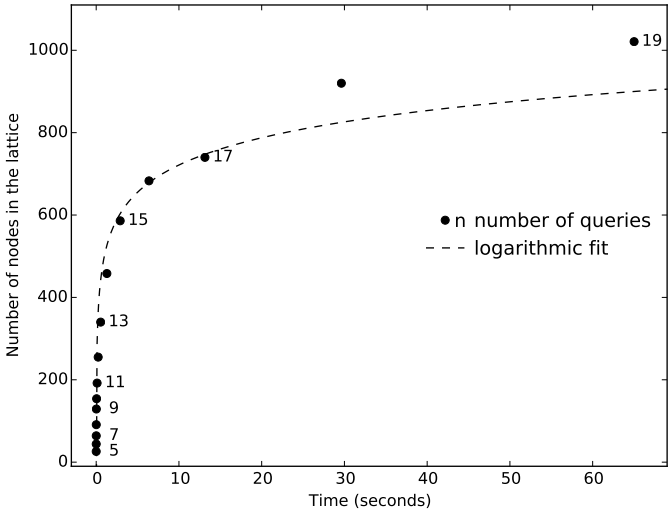

Figure 5.5: Time needed to generate the lattice nodes.

Datasets of 5 to 13 queries produce lattices almost instantly, in less than one second. Then, the time needed increases with an exponential tendency. The logarithmic fit curve for the generated points is represented as a dotted line and has the following definition:

$$y(x) = 95.35 * log(29.52 * x) + 179.16$$

Whereas the method is indeed based on combinations of initial queries, several reductions are applied. Firstly, the frequency of each query is calculated, leading to representing only unique queries. In cases when frequency values higher than 1 are detected,

these are labeled and propagated in the new classes of services generated. Secondly, when the vector of fuzzy values representing a query is the minimum of another vector, numerous nodes are generated, that are identical; again, only unique new classes of services are included in the lattice, which is another reduction opportunity scenario. Thirdly, queries in real world datasets do not tend to be thoroughly different, and the more similar they are, the fewer the lattice nodes. Therefore, despite its supposedly exponential character, our approach performs better than 2^n. The combinations initially computed by Algorithm 2 only concern the labels of queries and the actual calculations are only performed on the nodes that qualify to be part of the lattice.

The reductions applied lead to high delta (Δ) values between the number of combinations generated mathematically and the actual number of nodes needed to build the lattice. This is shown in Figure 5.6, that presents the discrepancy between the number of lattice nodes (gray) and the combinations (black), for 5, 10, 15 and 20 queries, respectively. For instance, for a dataset of 15 queries, Δ =1'047'464. The exponential fit curve for the middle points of Δ values is shown in red, and has the definition:

$$y(x) = 0.54 * e^{0.69*x}$$

The exponential character of this fit curve demonstrates that while the time needed to generate the lattices grows fast, the approach performs better than standard exponential, not needing all the mathematical combinations. When the input datasets contain

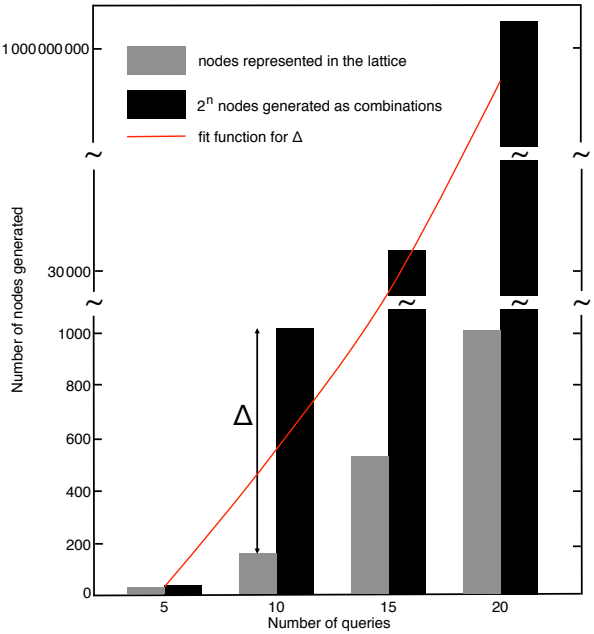

Figure 5.6: Lattice nodes vs. combinations.

less than 50 unique queries, the computing time remains in the range of hours, e.g., for 25 queries, without any pre-processing and bundling, the needed time is 3.85h. Nevertheless, as soon as bundling is applied, the time can drop drastically, depending on the value of the degree of similarity. This efficiency issue is known in the field of concept lattices, and various solutions have been investigated to mitigate it. For instance, Kumar and Srinivas [KS10b] apply k-means clustering for lattice reduction, defining a number of desired clusters k. However, they only focus on

the clusters resulted in their analysis, not mentioning anything about the performance gain. In our context, we can conclude that the time required for the data analysis is still much shorter than that of any other existing requirements elicitation techniques, even when several hours of computation are needed. Moreover, the results reported are obtained on our rather limited desktop machine. Algorithm 2 is principally parallelizable, and running it on a multi-core cluster should yield better results, decreasing the computation time proportionally to the number of cores used. However, this is subject to future work.

Automated Elicitation (R3)

As far as automation is concerned, our Galois lattices approach for cloud services is different from all the existing requirements elicitation techniques in that data collection is exclusively automated. The needs are gathered from (potential) consumers in a passive and unobtrusive way. Moreover, our technique is tool-supported [TKG15b], such that most of the data analysis is automated. As shown in Figure 5.2, the StakeCloud Tool performs numerous activities, such as computing bundles of similar queries. Whereas the new classes of services are automatically generated, while performing the analysis in A10, providers can perform a manual what-if investigation to dynamically simulate what happens when only one or a few features are varied, how these impact the general clustering, or zoom in specific parts of the lattice, to analyze the best ideas for new services.

For instance, Figure 5.7 shows a sample scenario in our tool. The top right panel contains the input dataset loaded, consisting of 20 distinct queries. The corresponding Galois lattice is displayed in the main panel, where the topmost element is the supremum for the entire lattice, and the first level in the hierarchy is composed of individual queries (circles, e.g., *(6)*) and bundles (rounded rectangles, e.g., *(2)*) generated based on a degree of similarity of 25 percent. The remaining nodes are classes of services generated automatically, as infima of the elements in the first hierarchy level.

Upon selecting three latice nodes displayed in gray (the bundle *(2)* and the individual queries *(1)* and *(4)*), the user can immediately visualize the corresponding supremum (*[]*, in green) and infima elements (red). The numerical fuzzy values of these are also displayed in the right central panel. As explained in Section 5.2.2 A7, the infima elements are the main candidates to evaluate when the provider is interested in satisfying consumers with searches such as *(1)*, *(2)* and *(4)*. For this, the user can select from five different criteria of analysis from the drop-down menu in the bottom right panel, e.g., analyze to what extent individual features are accomplished by selected nodes or their suprema/infima. Assuming the cloud provider is interested in satisfying as many as possible features fully for the selected queries, he selects the suprema/infima full satisfaction analysis. The tool automatically generates the graph displayed in the bottom right corner of the tool window, showing the results. The graph allows multiple plotting options, the scale can be changed and zooming capabilities are also embedded.

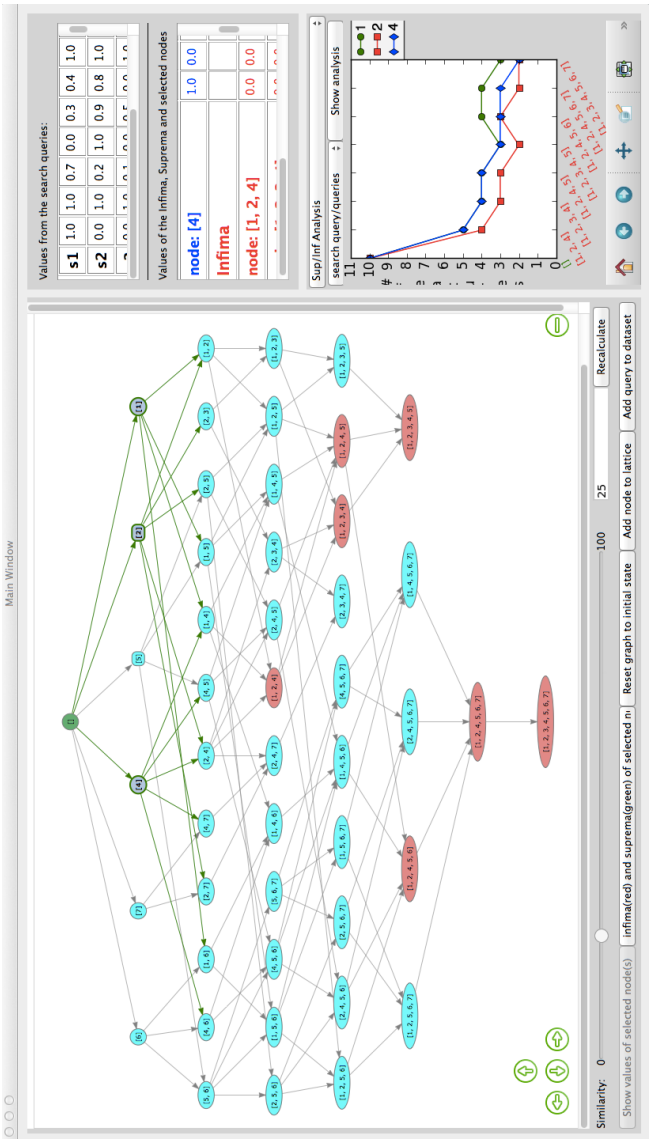

Figure 5.7: Tool screenshot - sample scenario.

As a next step, the user can decide to choose other analysis criteria, look into the individual queries composing the bundle *(2)*, or regenerate the lattice by changing the degree of similarity (as shown by the loops in Figure 5.2). Moreover, there is the option of adding new queries to the dataset. Lastly, the approach allows the user to model his/her existing offerings as vectors of fuzzy values, and then to search if these are among the lattice nodes. If they are found, they are highlighted, which helps cloud providers to see what queries they can satisfy with their current services, or how they could mildly enhance them to satisfy larger populations. In case their current services are not found among the nodes, a new lattice including them is generated, showing their relations to the other lattice elements.

Therefore, through the tool support provided, we propose our approach as a requirements elicitation technique that allows the automation of requirements acquisition to a large extent, and puts at cloud providers' disposal means for analyzing the automatically generated visualizations.

5.3.6 Consumers' Heterogeneity, Geographical Distribution and Volatile Requirements

In this section, we show how our approach addresses R1, R4 and R5, by enabling the elicitation of requirements from globally distributed audiences, whose needs are volatile.

Wide and Heterogeneous Audiences (R1)

The advanced searches needed by our fuzzy Galois lattices technique are always conducted on marketplaces websites. Therefore, our approach allows any number of consumers from virtually anywhere to input their needs for services in a completely asynchronous way. Moreover, according to the interviewed cloud providers [TSG13], performing advanced searches for services is among the most frequent methods used by both individual consumers and businesses to select cloud solutions suitable for their needs.

When generating the datasets used for experimentation, we took into account the heterogeneity aspect. Therefore, the queries used are highly heterogeneous, while they still follow the constraints posed by the features in Table 5.1. As a metric for heterogeneity, we use the standard deviation:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

In our case, N is the number of queries, x_i, $i = 1, N$ represent the individual queries, and μ is the mean of the queries in the dataset. We used the standard deviation for measuring the amount of dispersion from the average for the queries in our three datasets, and obtained the following results. For the dataset composed of 250 distinct queries, $\sigma_{250} \simeq 0.324$, for the 500 queries, $\sigma_{500} \simeq 0.322$,

and for the 1000 queries, $\sigma_{1000} \simeq 0.319$. All these values round to the value of 0.32, indicating that all the queries used in our tests are well spread, therefore heterogeneous.

Remote Application (R4)

Our approach is search-based, which means that it can be applied for any consumers, located anywhere, including those who are not physically reachable.

According to Use Case 3.9 defined for cloud computing by NIST (American National Institute of Standards and Technology)[8], "a cloud-user makes a structured capability or capacity or price request to one or several cloud-providers and receives a structured response that can be used as input to drive service decisions". This use case describes exactly the paradigm our approach is built on: the remote request for cloud service capabilities, upon which consumers get matching results from the marketplace. Such remote requests are the queries used as input, enabling the remote application of our approach. Moreover, remote analysis of the output of our method is also possible.

For instance, in the example shown in Figure 5.7, the automatically generated bottom right graph represents the number of features from queries *(1)* (green) and *(4)* (blue) and bundle *(2)* (red) that are fully satisfied by their supremum and infima. The information displayed in such graphs is available to the cloud provider company

[8] $http://www.nist.gov/itl/cloud/3\_9.cfm$ Accessed: June 2015.

without the need to send its consultants overseas. For example, the cloud provider representative can observe that classes of services like *(1,2,4,5,6)* and *(1,2,4,5,6,7)* would both fully satisfy four features for query *(1)* and three features for query *(4)*. However, *(1,2,4,5,6,7)* satisfies one feature less than *(1,2,4,5,6)* for bundle *(2)*. Nevertheless, it has the advantage of satisfying also query *(7)*, which belongs to the composing set. In this case, it is worth studying what is more valuable for the cloud provider: satisfying bundle *(2)* to a larger extent, or query *(7)*. The analysis can continue until the provider has obtained enough information to make a decision what service(s) are worth launching, e.g., by unbundling query *(2)* to analyze the queries it consists of, generating the graphs that visualize to what extent query *(7)* can be satisfied by the given infima, generating the graphs that show also the extent to which the features of the selected nodes are partially satisfied, etc. For another analysis example, please refer to [TG14]. This analysis is performed exclusively remotely, based on the input dataset.

Given its unobtrusive character, this technique is also suitable when consumers are not able to describe their requirements easily in an interview or a workshop, and can thus be used to complement other elicitation methods.

Volatile Requirements (R5)

Volatile requirements, i.e. requirements that change while being elicited, analyzed, validated and/or implemented, represent a recognized challenge for requirements elicitation methods. In contrast

to the existing techniques, where extensive time is allocated to first gathering requirements which are then analyzed, our approach enables a continuous elicitation process. Cloud consumers' data is collected continuously as advanced search queries on marketplaces, and can be instantly fed as input to our tool-supported approach. This way, volatile requirements can be monitored in an uninterrupted fashion. Moreover, due to the remote character of our method, this is done at virtually no additional cost. This feature is particularly fitting with the agile character of most cloud provider companies, which use fast development cycles and have rather short times to market.

Furthermore, our approach can be used for trend monitoring. For instance, it can calculate the mean for selected features of a series of datasets over a period of time. If it is detected that the mean value shifts steadily over that period of time, this represents a hint that the feature might follow that trend also in the future. For example, if the mean for the feature "storage" increases by 0.5GB every quarter for two years, this may indicate that a growth should be expected also in the following year(s). Conversely, if the frequency of a binary feature such as "AES encryption" decreases over a given series of datasets, this may be an indication that this features may be replaced by another, or consumers simply do not want it any more.

5.3.7 Threats to Validity

As far as construct validity is concerned, we tried to avoid evaluation apprehension by ensuring the product managers contacted

that all the information is anonymized and used exclusively for research purposes. We also mitigated hypothesis guessing by not giving them any details about our approach. Internal validity is threatened by the fact that we generated the datasets ourselves. However, we constructed them respecting the constraints of the features encoded and starting from a real world dataset. This way, we reduced the possible causal relationship between treatment and outcome. The fact that the datasets are self-generated concerns the external validity, i.e. generalizing results to industrial practice in particular. However, we attempted to build representative queries, as described above.

5.4 Related Work

From the early 2000s, researchers observed that requirements engineering also needs to consider distributed [DZ03] and asynchronous settings [GS05], and this currently extends to the cloud context. However, due to its collaboration-intensive and time-consuming nature, requirements elicitation becomes difficult in the cloud [DZ03, Dau00].

As far as *dedicated cloud requirements elicitation methods* are concerned, there has been some advancement during the recent years. For instance, frameworks focusing on the supply-demand relation have been designed [Liu12], Sun et al. developed a hybrid fuzzy framework for helping cloud consumers to select services that match their needs when their requests are uncertain [SDH+14],

and management systems for requirements ensuring QoS have been developed [VS11]. Moreover, researchers looked into methods for eliciting particular types of requirements, e.g., legal [BFS12] or security [BHC+13]. Still, these are only niche recommendations and no comprehensive clear solution exists, addressing the cloud-specific requirements elicitation challenges.

As far as *distributed requirements elicitation* is concerned, Lloyd et al. conducted a study [LRA02] on the effectiveness of elicitation techniques in distributed requirements engineering, concluding that synchronous elicitation approaches are generally more effective than asynchronous ones. Lim et al. [LQF10] present ideas on asynchronous and distributed stakeholder identification, assuming that key stakeholders are known, and further users can be identified based on domain knowledge. However, such approaches do not easily extend to the cloud context, since the audience for services is most often unknown and globally distributed.

Tuunanen [Tuu03] addresses the problem of reaching and involving wide audience end-users, or users who are not within organizational reach. He argues that traditional techniques do not provide adequate solutions and presents methods which could potentially fill this gap (e.g., EasyWinWin). However, none of these methods has been successfully used on a large scale for distributed elicitation so far. Moreover, research on EasyWinWin by Kukreja et al. [KB12] promises to provide support for distributed settings, but only focuses on stakeholders within organizational reach.

Another challenge of requirements elicitation in the cloud is the continuous change of consumer needs [Her07]. Consequently, various

wiki approaches have been implemented, to provide a time-efficient possibility for updating and eliciting requirements. For instance, Decker et al. developed a wiki-based solution that enables stakeholders' participation in RE [DRR+07], Solis and Ali's spatial hypertext wiki focuses on distributed teams [SA10], whereas Liang et al. [LAC09] and Lohmann et al. [LRAZ08] exploit semantic annotation wikis. However, wiki-based methods generally assume that stakeholders are at least identifiable, which is not the case in a cloud context.

Studies from the field of web-information systems by Yang and Tang [YT03] reveal that the elicitation needs regarding Internet-based systems are also rather different from those of traditional systems (e.g., due to higher user diversity). Moreover, most existing requirements elicitation methods can only deal with a limited number of stakeholders [Her07]. The number of potential cloud service consumers may often go beyond what traditional methods can handle [BHC+13], and no real solutions addressing this elicitation issue have been developed so far. Market-driven techniques [KDNoD+03], which are usually employed when it is impossible to consider individual consumers, prove to be rather limited in the cloud, due to the lack of specific localized markets.

Work in data mining, machine learning and particularly *recommender systems* [AT05] also addresses the problem of extracting value from search data. For example, search-based and collaborative techniques can make personalized online product recommendations [AXG11], and user feedback has been used to rank

various products [LPP08]. Throughout the recent years, recommender systems [RV97] (e.g., probability-based collaborative filtering [HKTR04]) and clustering data mining methods [Ber06] have been heavily used for marketing purposes, to suggest similar products in e-commerce systems, or to segment populations. However, to our knowledge, such techniques have never been adapted or utilized for requirements elicitation in the cloud.

Furthermore, data analysis methods that model consumers' needs have not been explored for the purpose of cloud requirements elicitation so far. Whereas there is an extensive body of research in the field of Galois concept lattices, most researchers exclusively focus on the mathematical implications of such graphs, and do not apply them in a practical context. There are only a few examples of attempts where lattices are used to identify objects or concepts in given datasets [vDK99, GMA95], or applied to browsing retrieval [CR96]. Most existing lattice-related work focuses on clustering opportunities with Galois lattices from a purely theoretical perspective [PRSV02].

To summarize, the existing elicitation techniques, even when adapted, are mostly unsuitable for the cloud, and can support cloud service providers only to a limited extent in their requirements elicitation processes. Moreover, the existing data mining and lattice approaches have not been applied in the requirements acquisition context so far, to our knowledge, leaving the issue of dedicated cloud requirements elicitation techniques unsolved.

5.5 Conclusion and Future Work

In this paper, we proposed a new approach to requirements elicitation for cloud services, that builds fuzzy Galois lattices based on consumers' advanced search queries. We evaluated it against five main cloud providers' requirements and showed how it addresses them: it can gather needs from wide and heterogeneous audiences whose requirements are volatile, automates data analysis and can be applied remotely, taking less time than traditional elicitation methods. Our approach is best-suited for the early elicitation phase and for monitoring market trends. It can be succeeded by more in-depth requirements elicitation with complementary methods such as prototyping and large-scale online experiments. Although our approach was natively built for cloud contexts, we foresee that it can be applied successfully also in other domains that exhibit similar properties and where queries can be collected in a similar fashion, e.g., in traditional web-service and service-oriented systems. However, we have not investigated this usage scenario so far, but it is subject to future work.

A limitation of our approach is that it assumes consumers provide values for all the features specified as advanced search criteria. Currently, we ignore the queries with values of zero for all features (outliers) and allocate default values when no values are assigned. We are now working on implementing the maximum likelihood estimation, which uses the available data to compute maximum likelihood estimates. Moreover, we plan to improve the time performance and method scalability by using MapReduce to

compute the minima values in Algorithm 2. In addition, we are currently working on extending the approach to work when particular features have weights denoting their importance. Naturally, adoption issues related to users' data privacy may occur when our method is deployed in practice and some time may be needed until cloud providers get used to the workflow. Therefore, we plan to apply our elicitation method in real world settings, with cloud providers and their datasets, to discover any potential issues and demonstrate its actual impact and practical use.

Chapter 6
Conclusions

6.1 Summary and Achievements

Successful services and products strongly depend on appropriate stakeholder requirements communication methods and tools, and cloud services are no exception to this general rule. However, due to the challenges posed by the cloud paradigm, existing requirements elicitation techniques have become virtually obsolete, leading to an evident need for dedicated cloud requirements communication approaches. This motivated us to first investigate cloud providers' current state of practice concerning the elicitation activity and then to develop StakeCloud, a novel cloud requirements communication approach.

We summarize our work and achievements by answering the research questions introduced in Section 1.3.

Research Question 1: How do cloud service providers elicit consumer requirements?

In Chapter 2, we presented an exploratory study on how 19 cloud provider companies from ten countries perform the requirements elicitation activity. The semi-structured interviews we conducted revealed that the most popular elicitation techniques are interviews, questionnaires, documentation analysis, surveys and prototyping. These methods were regarded as difficult to nearly impossible to apply in most cloud cases, which led to providers using ad-hoc methods to learn about their (potential) consumers' needs. Moreover, we found that cloud providers' satisfaction level with regard to applying existing elicitation methods is generally low to medium. This dissatisfaction is often caused by the evolution history of the company, i.e. the transition from a traditional to a cloud service provider was made without taking the elicitation aspect into account, only focusing on technical aspects.

Research Question 2: What features should a dedicated cloud requirements elicitation method have to help cloud providers understand their (potential) consumers' needs?

Besides reviewing the requirements elicitation methods that are most popular among cloud providers, the exploratory study described in Chapter 2 also showed what kind of methods the cloud paradigm calls for. They should fit for heterogeneous and wide audiences, take less time than existing techniques and should allow providers to apply them remotely. Furthermore, automation is needed to a larger extent for eliciting consumers' requirements, which are often volatile.

Research Question 3: How can (potential) consumers' advanced search queries for cloud services be used to infer requirements?

Chapter 3 introduced a novel approach for inferring requirements from (potential) cloud consumers' advanced search queries. It has its roots in Galois theory and builds fuzzy Galois lattices based on what consumers search for. The infima elements computed represent compromise services that satisfy the initial queries to limited extents, which can be adjusted depending on the goals and preferences of cloud providers. Using a running example, we showed how StakeCloud, our conceptual fuzzy Galois approach, supports the requirements for a dedicated cloud elicitation method identified during the exploratory study.

Research Question 4: How does our approach meet cloud service providers' needs for a dedicated cloud requirements elicitation method?

Upon implementing and extending the conceptual StakeCloud approach, we proceeded to evaluate it. For this, we analyzed to what extent cloud providers' requirements for a new dedicated cloud elicitation method are met by StakeCloud. The results are communicated in Chapter 5. Firstly, we investigated whether automation and time efficiency were achieved. For this, we analyzed the time performance of the similarity classifier embedded in our solution, using datasets ranging from 250 to 1000 queries. While these were self-generated, they fully maintained the characteristics of the smaller original datasets provided by cloud companies,

which acted as a starting point. According to our results, all clusters for datasets of up to 1000 queries could be generated in less than two seconds, irrespective of cluster sizes. Moreover, despite its formally exponential character, our approach always performed significantly better than $\mathcal{O}(2^n)$ due to the reductions applied and the particular features of advanced search queries for services (e.g., high frequencies and similarity). Secondly, thanks to the implemented tool, automation was achieved to a large extent, since lattices are automatically generated, as well as all the analysis graphs needed during the decision-making process. Thirdly, we showed that the StakeCloud approach fits for wide and heterogeneous audiences: the average standard deviation for our datasets was 0.32, indicating the queries used in the experiments were well spread. Moreover, remote application is possible since the method is exclusively based on data collected online. Therefore, our approach also supports volatile requirements, since trends can be monitored over time and change requests can be identified at an early stage. In conclusion, we consider our research successful since a proof of concept of our approach was achieved, that solves the issues identified with RQ1, and the results of the evaluation showed that the requirements determined with RQ2 are met.

By answering these research questions, we demonstrated our Thesis Statement, showing that cloud consumers' advanced search queries can be used to infer new service requirements, such that cloud providers deliver solutions targeted at consumers' real needs. The StakeCloud approach supports cloud providers in choosing the most suitable compromise services to satisfy large populations of consumers with a minimum set of requirements implemented,

to reach an optimum solution. Therefore, our approach goes beyond plain matching and simple statistics, since the lattice infima elements represent newly generated classes of services and novel combinations of feature values. These cannot be inferred by counting frequencies in the initial datasets or using multiple-criteria decision analysis based on standard statistical methods.

According to Wieringa and Heerkens [WH06], the relevance criteria for a solution consists of three elements: (i) novelty of the solution, (ii) relevance for classes of world problems, and (iii) relevance for theory. We evaluate our solution as relevant, since (i) it is a novel approach, based on a mathematical model that has never been exploited for this purpose before. Furthermore, (ii) it contributes to solving a significant real world problem: eliciting consumer requirements in cloud settings. Finally, (iii) our solution is relevant for theory: on the one hand, it builds on top of existing research and best practices. On the other hand, it extends them with a new model for requirements analysis and opens numerous new research directions.

6.2 Outlook

StakeCloud represents one of the first dedicated approaches aimed at supporting cloud service providers in eliciting and understanding their consumers' needs. Despite the progress made in the direction of building a dedicated cloud requirements communication approach, there are still related ideas and areas worth exploring in

the future. This section discusses some current limitations of the StakeCloud method and suggests potential future paths.

The current version of the StakeCloud approach benefits from sophisticated tool support. The proof of concept we built offers numerous features that cloud provider representatives can use during their analysis and decision-making processes. However, the choice of features, options and parameters may not always be intuitive, thus leading to a steep learning curve and potentially a need for extended adoption periods. Therefore, one possible future direction consists of embedding an *artificial intelligence-powered wizard* in the existing tool that guides the cloud provider representative through the entire requirements analysis process. For instance, (s)he could first input key stakeholders' features or define his/her company's unique selling points in terms of service characteristics, and then the wizard could recommend what existing visualizations are appropriate and the corresponding parameters. Moreover, concrete formalisms could be developed for calculating the satisfaction level for particular combinations of service features, based on cloud providers' input, when making the analysis.

A further way of enhancing the usability of the tool-supported StakeCloud approach consists of making the *frequency of each lattice node more easily readable*. In the current version, in case a particular query appears more than once in the initial dataset, it is represented as a node which has a frequency equal to the number of occurrences in the dataset. The frequency can be read by the StakeCloud user when hovering over the node. This visualization could be enhanced by, e.g., increasing the size of

the nodes such that they are proportional to their frequencies, or using a color gradient. Additionally, a similar visualization could be used for displaying clusters, where their size or color depends on the number of underlying queries. This information could be further propagated and displayed on the other lattice nodes representing service candidates that satisfy high frequency queries or large clusters.

Since our solution was built following the pragmatic paradigm, to solve a real world problem, it is essential to conduct *further evaluations* using cloud provider companies' datasets. The services they decide to release based on the recommendations made by our approach should then be monitored over time. The real success and results of StakeCloud can only be seen in the future, after such solutions have been launched and related usage data and feedback have been collected. StakeCloud was natively designed to be used by marketplaces, on the advanced search queries data gathered. Therefore, a next step would be to embed it in existing marketplace platforms, such that the requirements inferred can then be sent to the cloud providers that sell their solutions through these marketplaces. In addition, large-scale studies could be conducted, e.g., to *compare consumers' initial queries with the services purchased* and to analyze their *level of satisfaction* after a defined period of usage. Such studies may offer reliable hints regarding the compromise consumers are willing to make.

Furthermore, to make the approach more accessible, the current cross-platform desktop application could be accompanied by a *Web interface*. This would not have to include all the existing

features, but could resemble a dashboard that gives cloud providers a general overview of the queries, e.g., for trend monitoring.

Currently, the data used as input by StakeCloud is represented by fuzzy vectors, thus purely structured data. One potential research direction would be to run our approach also with other data that are first mined from *natural language descriptions*, such as app reviews or documentation resulting from interviews and workshops. One challenge when exploiting such data may be their incompleteness, i.e. values may not be available for all the predefined features. However, this issue can be solved by implementing, e.g., the maximum likelihood estimation which uses the available data to populate empty fields.

Another input data-related extension of StakeCloud could consist of modeling and analyzing advanced search queries that also include information about the *weights* of particular features, and not only the values given for each feature. For instance, a consumer may regard the uptime of 100% of a service as critical and not want to compromise on this, the storage size of 1TB may be a nice to have, whereas (s)he may not care about the other features. The StakeCloud approach includes a preliminary version of such a weights-driven analysis, which allows manipulating datasets of queries that specify the weight for each feature. Depending on their weights, features can be: "must-have" features, "nice to have" features and "don't care" features, and each query can contain one "must-have" feature at most and any number of features of the other two types. The StakeCloud approach propagates these weights in the lattice, to the compromise services generated, such that infima

elements always include the consumer-imposed weights. Although we implemented this concept, we have not conducted a thorough evaluation due to the lack of real-world datasets including weights and the limited interest of our cooperating companies in such an extension. Nevertheless, further research in this direction has good potential for valuable results.

Originally, StakeCloud was designed and built for the cloud domain, to support cloud service providers. Nevertheless, the nature of its conceptual model allows its application also in *other domains*. The only constraint consists of the input data representation and availability. As long as consumers' queries can be modeled as fuzzy vectors of feature values and such advanced search queries are available, our approach can be applied in any other domain. Consequently, it would be interesting to also evaluate how the StakeCloud method performs outside the cloud context.

Finally, it should be noted that our approach represents an early elicitation phase method which can naturally be followed by other existing, more in-depth elicitation techniques with selected stakeholders. The requirements inferred with StakeCloud can further be formalized and transformed into *specifications*, for instance using existing tool-supported methods such as the one proposed by Li et al.[LHB+15].

In summary, we believe that StakeCloud not only represents one step towards solving the requirements communication problem in cloud settings, but also a starting point for new inspiring research directions in the areas of requirements analysis and modeling, monitoring and decision-making support.

Bibliography

[ACL+12] Rahul Akolkar, Tom Chefalas, Jim Laredo, Perng Chang-Shing, Anca Sailer, Frank Schaffa, Ignacio Silva-Lepe, and Tao Tao. Towards cloud services marketplaces. In *8th International Conference and Workshop on Systems Virtualization Management*, pages 179–183. IEEE, 2012.

[AFG+09] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the clouds: a Berkeley view of cloud computing. Technical report, University of California, Berkeley, 2009.

[APC06] Carina Alves, Silvia Pereira, and Jaelson Castro. A study in market-driven requirements engineering. Technical report, Universidade Federal de Pernambuco, 2006.

[AT05] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 17(6):734–749, 2005.

[AW05] Aybüke Aurum and Claes Wohlin. *Engineering and managing software requirements*. Springer-Verlag Berlin, 2005.

[AXG11] Noraswaliza Abdullah, Yue Xu, and Shlomo Geva. Integrating collaborative filtering and search-based techniques for personalized online product recommendation. In *11^{th} IEEE International Conference on Data Mining Workshops (ICDMW'11)*, pages 711–718. IEEE, 2011.

[Bel99] Radim Belohlavek. Fuzzy Galois connections. *Mathematical Logic Quarterly*, 45(4):497–504, 1999.

[BELK10] Marina Berkovich, Sebastian Esch, Jan Marco Leimeister, and Helmut Krcmar. Towards requirements engineering for Software as a Service. *Multikonferenz Wirtschaftsinformatik 2010*, page 107, 2010.

[Ber06] P. Berkhin. A survey of clustering data mining techniques. In Jacob Kogan, Charles Nicholas, and Marc Teboulle, editors, *Grouping Multidimensional Data*, pages 25–71. Springer Berlin Heidelberg, 2006.

[BFS12] Kristian Beckers, Stephan Fassbender, and Holger Schmidt. An integrated method for pattern-based elicitation of legal requirements applied to a cloud computing example. In *7th International Conference on Availability, Reliability and Security (ARES'12)*, pages 463–472. IEEE, 2012.

[BHC+13] Kristian Beckers, Maritta Heisel, Isabelle Cote, Ludger Goeke, and Selim Güler. Structured pattern-based security requirements elicitation for clouds. In *8th International Conference on Availability, Reliability and Security (ARES'13)*, pages 465–474. IEEE, 2013.

[BJS+08] Nicolas Bettenburg, Sascha Just, Adrian Schröter, Cathrin Weiss, Rahul Premraj, and Thomas Zimmermann. What makes a good bug report? In *16th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 308–318. ACM, 2008.

[BK14] Azer Bestavros and Orran Krieger. Toward an open cloud marketplace: Vision and first steps. *IEEE Internet Computing*, 18(1):72–77, 2014.

[Bla10] Loraine Blaxter. *How to research*. McGraw-Hill Education United Kingdom, 2010.

[BYV+09] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud

computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599–616, 2009.

[CB95] Erran Carmel and Shirley Becker. A process model for packaged software development. *IEEE Transactions on Engineering Management (TEM)*, 42(1):50–61, 1995.

[Clo15] Cloudorado - Cloud Computing Comparison Engine. Available at: `https://www.cloudorado.com/`, Accessed: November 2015.

[CLPZ11] K. Selçuk Candan, Wen-Syan Li, Thomas Phan, and Minqi Zhou. At the frontiers of information and Software as Services. In *New Frontiers in Information and Software as Services*, pages 283–300. Springer, 2011.

[CND11] Sanjay Chakraborty, N. K. Nagwani, and Lopamudra Dey. Performance comparison of incremental k-means and incremental DBSCAN algorithms. *International Journal of Computer Applications*, 27(11):14–18, 2011.

[CPK10] Yanpei Chen, Vern Paxson, and Randy H. Katz. What's new about cloud computing security. *University of California, Berkeley Report No. UCB/EECS-2010-5*, 20(2010):1–8, 2010.

[CR96] Claudio Carpineto and Giovanni Romano. A lattice conceptual clustering system and its application to browsing retrieval. *Machine Learning*, 24(2):95–122, 1996.

[Cre13] John W. Creswell. *Research design: Qualitative, quantitative, and mixed methods approaches*. SAGE Publications, 2013.

[Dau00] A. Daugulis. Time aspects in requirements engineering: Or 'every cloud has a silver lining'. *Requirements Engineering (RE)*, 5(3):137–143, 2000.

[Deu15] Deutsche Börse Cloud Exchange. Available at: `https://cloud.exchange/en/`, Accessed: November 2015.

[DEW13] Klaus Denecke, Marcel Erné, and Shelly L. Wismath. *Galois Connections and Applications*, volume 565. Springer Science & Business Media, 2013.

[Dia14] Stephen L. Diamond. The next computing platform and cloud computing standards (Keynote). In *10th World Congress on Services (SERVICES'14)*. IEEE, 2014.

[DKP+03] Åsa D. Dahlstedt, Lena Karlsson, Anne Persson, Johan Natt och Dag, and Björn Regnell. Market-driven requirements engineering processes for software products - a report on current practices. In *The*

International Workshop on COTS and Product Software RECOTS, in conjunction with the 11th IEEE International Requirements Engineering Conference. IEEE, 2003.

[DRR+07] Björn Decker, Eric Ras, Jörg Rech, Pascal Jaubert, and Marco Rieth. Wiki-based stakeholder participation in requirements engineering. *IEEE Software*, 24(2):28–35, 2007.

[DZ03] Daniela E. Damian and Didar Zowghi. RE challenges in multi-site software development organisations. *Requirements Engineering (RE)*, 8(3):149–160, 2003.

[EKMS93] Marcel Erné, Jürgen Koslowski, Austin Melton, and George E. Strecker. A primer on Galois connections. *Annals of the New York Academy of Sciences*, 704(1):103–125, 1993.

[ESSD08] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. Selecting empirical methods for software engineering research. In *Guide to Advanced Empirical Software Engineering*, pages 285–311. Springer, 2008.

[FdS09] Carla Farinha and Miguel Mira da Silva. Focus groups for eliciting requirements in information systems development. In *UK Academy for Information Systems*, page 26, 2009.

[FdS11] Carla Farinha and Miguel Mira da Silva. Web-based focus groups for requirements elicitation. In *6th International Conference on Software Engineering Advances (ICSEA'2011)*, 2011.

[Fis00] Bernd Fischer. Specification-based browsing of software component libraries. *Automated Software Engineering*, 7(2):179–200, 2000.

[FR15] James Michael Ferris and Gerry Edward Riveros. Cross-cloud vendor mapping service in cloud marketplace, February 2015. US Patent 8 954 564.

[GL13] Stefanos Gritzalis and Lin Liu. Requirements engineering for security, privacy and services in cloud environments. *Requirements Engineering (RE)*, 18(4):297–298, 2013.

[Gli07] Martin Glinz. On non-functional requirements. In *15th IEEE International Requirements Engineering Conference (RE'07)*, pages 21–26. IEEE, 2007.

[Gli15] Martin Glinz. A Glossary of Requirements Engineering Terminology. Version 1.4 September 2012. International Requirements Engineering Board (IREB). Available at: `http://fileadmin.cs.lth.se/cs/Education/ETS170/IREB_CPRE_Glossary_14.pdf`. Accessed: October 2015.

[GMA95] Robert Godin, Rokia Missaoui, and Hassan Alaoui. Incremental concept formation algorithms based on Galois (concept) lattices. *Computational Intelligence*, 11(2):246–267, 1995.

[GS05] Paul Grünbacher and Norbert Seyff. Requirements negotiation. In *Engineering and Managing Software Requirements*, pages 143–162. Springer, 2005.

[GW07] Martin Glinz and Roel J. Wieringa. Stakeholders in requirements engineering. *IEEE Software*, 24(2):18–20, 2007.

[GWF97] Bernhard Ganter, Rudolf Wille, and Cornelia Franzke. *Formal concept analysis: mathematical foundations*. Springer-Verlag New York, 1997.

[HD03] Ann M. Hickey and Alan M. Davis. Elicitation technique selection: how do experts do it? In *11^{th} IEEE International Requirements Engineering Conference (RE'03)*, pages 169–178. IEEE, 2003.

[HEGM13] Abram Hindle, Neil A. Ernst, Michael W. Godfrey, and John Mylopoulos. Automated topic naming. *Empirical Software Engineering*, 18(6):1125–1155, 2013.

[Her07] James D. Herbsleb. Global software engineering: The future of socio-technical coordination. In *Future of Software Engineering*, pages 188–198. IEEE Computer Society, 2007.

[HKTR04] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TIS)*, 22(1):5–53, 2004.

[Hon95] E. Honour. *Principles of Commercial Systems Engineering.* 5th Annual International Symposium of the National Council on Systems Engineering, 1995.

[ID12] Iliana Iankoulova and Maya Daneva. Cloud computing security requirements: A systematic review. In *6th International Conference on Research Challenges in Information Science (RCIS'12)*, pages 1–7, May 2012.

[IG11] Martin Ivarsson and Tony Gorschek. A method for evaluating rigor and industrial relevance of technology evaluations. *Empirical Software Engineering*, 16(3):365–395, 2011.

[IH10] Bala Iyer and John C. Henderson. Preparing for the future: Understanding the seven capabilities of cloud computing. *MIS Quarterly Executive*, 9(2):117–131, 2010.

[Int15] Intel Cloud Finder.
Available at: `http://www.intelcloudfinder.com/`,
Accessed: November 2015.

[KAD10] Philip Koehler, Arun Anandasivam, and M. A. Dan. Cloud services from a consumer perspective. In *16th Americas Conference on Information Systems (AMCIS'10)*, pages 1–10, 2010.

[KADW10] Philip Koehler, Arun Anandasivam, M. A. Dan, and Christof Weinhardt. Customer heterogeneity and tariff biases in cloud computing. In *31st International Conference on Information Systems (ICIS'10)*, page 106, 2010.

[KB12] Nupul Kukreja and Barry Boehm. Process implications of social networking-based requirements negotiation tools. In *International Conference on Software and System Process (ICSSP'12)*, pages 68–72. IEEE, 2012.

[KBKF09] Eric Knauss, Olesia Brill, Ingo Kitzmann, and Thomas Flohr. Smartwiki: Support for high-quality requirements engineering in a collaborative setting. In *ICSE Workshop on Wikis for Software Engineering (WIKIS4SE'09)*, pages 25–35. IEEE, 2009.

[KDNoD+03] Lena Karlsson, Åsa Dahlstedt, Johan Natt och Dag, Björn Regnell, and Anne Persson. Challenges in market-driven requirements engineering - an industrial interview study. In *8th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03)*, pages 101–112. Essener Informatik-Beiträge, 2003.

[KHGSS12] Ali Khajeh-Hosseini, David Greenwood, James W. Smith, and Ian Sommerville. The cloud adoption toolkit: supporting cloud adoption decisions in the enterprise. *Software: Practice and Experience*, 42(4):447–465, 2012.

[KKLK05] Sari Kujala, Marjo Kauppinen, Laura Lehtola, and Tero Kojo. The role of user involvement in requirements quality and project success. In *13th IEEE International Conference on Requirements Engineering (RE'05)*, pages 75–84. IEEE, 2005.

[KMI13] Christos Kalloniatis, Haralambos Mouratidis, and Shareeful Islam. Evaluating cloud deployment scenarios based on security and privacy requirements. *Requirements Engineering (RE)*, 18(4):299–319, 2013.

[KS10a] Jaeyong Kang and Kwang Mong Sim. Cloudle: a multi-criteria cloud service search engine. In *IEEE Asia-Pacific Services Computing Conference (APSCC'10)*, pages 339–346. IEEE, 2010.

[KS10b] Aswani Ch. Kumar and S. Srinivas. Concept lattice reduction using fuzzy k-means clustering. *Expert Systems with Applications*, 37(3):2696–2704, 2010.

[Kuj03] Sari Kujala. User involvement: a review of the benefits and challenges. *Behaviour & Information Technology*, 22(1):1–16, 2003.

[LAC09] Peng Liang, Paris Avgeriou, and Viktor Clerc. Requirements reasoning for distributed requirements analysis using semantic wiki. In *4th IEEE International Conference on Global Software Engineering (ICGSE'09)*, pages 388–393. IEEE, 2009.

[LBRK10] Stefanie Leimeister, Markus Böhm, Christoph Riedl, and Helmut Krcmar. The business perspective of cloud computing: Actors, roles and value networks. In *18th European Conference on Information Systems (ECIS'10), Paper 56*, 2010.

[LHB+15] Feng-Lin Li, Jennifer Horkoff, Alexander Borgida, Giancarlo Guizzardi, Lin Liu, and John Mylopoulos. From stakeholder requirements to formal specifications through refinement. In *Requirements Engineering: Foundation for Software Quality (REFSQ'15)*, volume 9013 of *Lecture Notes in Computer Science*, pages 164–180. Springer, 2015.

[Lik32] Rensis Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 1932.

[Liu12] Lin Liu. *Software Reuse in the Emerging Cloud Computing Era: Goal-Based Requirements Elicitation for Service Reuse in Cloud Computing.* IGI Global, 2012.

[LKN+09] Alexander Lenk, Markus Klems, Jens Nimis, Stefan Tai, and Thomas Sandholm. What's inside the

cloud? An architectural map of the cloud landscape. In *ICSE Workshop on Software Engineering Challenges of Cloud Computing*, pages 23–31. IEEE Computer Society, 2009.

[LNH07] Sharman Lichtenstein, Lemai Nguyen, and Alexia Hunter. Issues in IT service-oriented requirements engineering. *Australasian Journal of Information Systems*, 13(1):176–191, 2007.

[LPP08] Tong Queue Lee, Young Park, and Yong-Tae Park. A time-based approach to effective recommender systems using implicit feedback. *Expert Systems with Applications*, 34(4):3055–3062, 2008.

[LQF10] Soo Ling Lim, Daniele Quercia, and Anthony Finkelstein. StakeNet: using social networks to analyse the stakeholders of large-scale software projects. In *22nd ACM/IEEE International Conference on Software Engineering (ICSE'10)*, pages 295–304. ACM, 2010.

[LRA02] Wesley James Lloyd, Mary Beth Rosson, and James D. Arthur. Effectiveness of elicitation techniques in distributed requirements engineering. In *IEEE Joint International Conference on Requirements Engineering*, pages 311–318. IEEE, 2002.

[LRAZ08] Steffen Lohmann, Thomas Riechert, Sören Auer, and Jürgen Ziegler. Collaborative development of

knowledge bases in distributed requirements elicitation. In *Walid Maalej, Bernd Brügge (eds.). Software Engineering 2008 - Workshop Volume. Proc. GI Conference Software Engineering 2008, Munich, Germany. GI Lecture Notes in Informatics*, volume 122, pages 22–28, 2008.

[LTM+] Fang Liu, Jin Tong, Jian Mao, Robert B. Bohn, John V. Messina, Mark L. Badger, and Dawn M. Leaf. NIST Cloud Computing Reference Architecture. National Institute of Standards Special Publication (NIST SP) 500-292, September 2011. Available at: `http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909505`, Accessed: November 2015.

[MCN92] John Mylopoulos, Lawrence Chung, and Brian Nixon. Representing and using nonfunctional requirements: A process-oriented approach. *IEEE Transactions on Software Engineering (TSE)*, 18(6):483–497, 1992.

[MG11] Peter Mell and Tim Grance. The NIST definition of cloud computing. *NIST - National Institute of Standards and Technology, Special Publication 800 - 145*, pages 1–7, 2011.

[MIKG13] Haralambos Mouratidis, Shareeful Islam, Christos Kalloniatis, and Stefanos Gritzalis. A framework to support selection of cloud providers based on security and privacy requirements. *Journal of Systems and Software (JSS)*, 86(9):2276 – 2293, 2013.

[MK06] Noella Mackenzie and Sally Knipe. Research dilemmas: Paradigms, methods and methodology. *Issues in Educational Research*, 16(2):193–205, 2006.

[MR96] Neil A. M. Maiden and Gordon Rugg. ACRE: selecting methods for requirements acquisition. *Software Engineering Journal*, 11(3):183–192, 1996.

[MTG11] Benedikt Martens, Frank Teuteberg, and Matthias Gräuler. Design and implementation of a community platform for the evaluation and selection of cloud computing services: a market analysis. In *European Conference on Information Systems (ECIS'11), Paper 215*, 2011.

[NE00] Bashar Nuseibeh and Steve Easterbrook. Requirements engineering: a roadmap. In Anthony Finkelstein, editor, *Special Track of the 22nd ACM/IEEE International Conference on Software Engineering: Conference on the Future of Software Engineering:*, pages 35–46. ACM, 2000.

[NER00] Bashar Nuseibeh, Steve Easterbrook, and Alessandra Russo. Leveraging inconsistency in software development. *Computer*, 33(4):24–29, 2000.

[NK09] Humaira Naz and Muhammad Nadeem Khokhar. Critical requirements engineering issues and their solution. In *International Conference on Computer Modeling and Simulation (ICCMS'09)*, pages 218–222. IEEE, 2009.

[Pat90] Michael Quinn Patton. *Qualitative evaluation and research methods*. SAGE Publications, 1990.

[Poh10] Klaus Pohl. *Requirements engineering: fundamentals, principles, and techniques*. Springer, 2010.

[Pot95] Colin Potts. Invented requirements and imagined customers: requirements engineering for off-the-shelf software. In *2^{nd} IEEE International Symposium on Requirements Engineering*, pages 128–130. IEEE, 1995.

[PRSV02] Nathalie Pernelle, Marie-Christine Rousset, Henry Soldano, and Véronique Ventos. Zoom: a nested Galois lattices-based system for conceptual clustering. *Journal of Experimental & Theoretical Artificial Intelligence*, 14(2-3):157–187, 2002.

[Ram13] Muthu Ramachandran. Business requirements engineering for developing cloud computing services. In Zaigham Mahmood and Saqib Saeed, editors, *Software Engineering Frameworks for the Cloud Computing Paradigm*, Computer Communications and Networks, pages 123–143. Springer London, 2013.

[Rob02] Colin Robson. *Real world research: A resource for social scientists and practitioner-researchers*, volume 2. Blackwell Oxford, 2002.

[RV97] Paul Resnick and Hal R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[RZWT12] Jonas Repschlaeger, Ruediger Zarnekow, Stefan Wind, and Klaus Turowski. Cloud requirement framework: Requirements and evaluation criteria to adopt cloud solutions. In *European Conference on Information Systems (ECIS'12), Paper 42*, 2012.

[SA10] Carlos Soli and Nour Ali. Distributed requirements elicitation using a spatial hypertext wiki. In *5th IEEE International Conference on Global Software Engineering (ICGSE'10)*, pages 237–246. IEEE, 2010.

[Saw00] Peter Sawyer. Packaged software: challenges for RE. In *Sixth International Workshop on Requirements Engineering: Foundations of Software Quality (REFSQ'00)*. Essener Informatik-Beiträge, 2000.

[Sch04] Barry Schwartz. The paradox of choice: Why less is more. *New York: Ecco*, 2004.

[SDH+14] Le Sun, Hai Dong, Farookh Khadeer Hussain, Omar Khadeer Hussain, Jiangang Ma, and Yanchun Zhang. A hybrid fuzzy framework for cloud service selection. In *IEEE International Conference on Web Services (ICWS'14)*, pages 313–320. IEEE, 2014.

[SGM10a] Norbert Seyff, Florian Graf, and Neil Maiden. End-user requirements blogging with iRequire. In *32nd ACM/IEEE International Conference on Software Engineering (ICSE'10)*, pages 285–288. ACM, 2010.

[SGM10b] Norbert Seyff, Florian Graf, and Neil Maiden. Using mobile RE tools to give end-users their own voice. In *18th IEEE International Requirements Engineering Conference (RE'10)*, pages 37–46. IEEE, 2010.

[SK98] Ian Sommerville and Gerald Kotonya. *Requirements engineering: processes and techniques*. John Wiley & Sons Inc., 1998.

[SLAM13] Vítor E. Silva Souza, Alexei Lapouchnian, Konstantinos Angelopoulos, and John Mylopoulos. Requirements-driven software evolution. *Computer Science - Research and Development*, 28(4):311–329, 2013.

[SMLD97] Houari A. Sahraoui, Walcklio Melo, Hakim Lounis, and François Dumont. Applying concept formation methods to object identification in procedural code. In *12th IEEE International Conference in Automated Software Engineering (ASE'97)*, pages 210–218. IEEE, 1997.

[SR99] Michael Siff and Thomas Reps. Identifying modules via concept analysis. *IEEE Transactions on Software Engineering (TSE)*, 25(6):749–768, 1999.

[SS97] Ian Sommerville and Pete Sawyer. *Requirements engineering: a good practice guide*. John Wiley & Sons Inc., 1997.

[STC+15] Norbert Seyff, Irina Todoran, Kevin Caluser, Leif Singer, and Martin Glinz. Using popular social network sites to support requirements elicitation, prioritization and negotiation. *Journal of Internet Services and Applications (JISA)*, 6(1):1–16, 2015.

[SW11] Holger Schrödl and Stefan Wind. Requirements engineering for cloud computing. *Journal of Communication and Computer*, 8(9):707–715, 2011.

[TBvdZ04] Jos J. M. Trienekens, Jacques J. Bouman, and Mark van der Zwan. Specification of service level agreements: Problems, principles and practices. *Software Quality Journal*, 12(1):43–57, 2004.

[TG12] Irina Todoran and Martin Glinz. Towards bridging the communication gap between consumers and providers in the cloud. In *10th Working IEEE/IFIP Conference on Software Architecture (WICSA/ECSA'12)*, pages 78–79. ACM, 2012.

[TG14] Irina Todoran and Martin Glinz. Quest for requirements: Scrutinizing advanced search queries for cloud services with fuzzy Galois lattices. In *IEEE 10th World Congress on Services (SERVICES'14)*, pages 234–241. IEEE, 2014.

[TKG15a] Irina Todoran Koitz and Martin Glinz. A fuzzy Galois lattices approach to requirements elicitation for cloud services. *IEEE Transactions on Services Computing (TSC). DOI: 10.1109/TSC.2015.2466538. Published online*, 2015.

[TKG15b] Irina Todoran Koitz and Martin Glinz. StakeCloud Tool: From cloud consumers' search queries to new service requirements. In *23rd IEEE International Requirements Engineering Conference (RE'15)*, pages 284–285. IEEE, 2015.

[Tod12] Irina Todoran. StakeCloud: Stakeholder requirements communication and resource identification in the cloud. In *Doctoral Symposium of the 20^{th} IEEE International Requirements Engineering Conference (RE'12)*, pages 353–356. IEEE, 2012.

[TSG13] Irina Todoran, Norbert Seyff, and Martin Glinz. How cloud providers elicit consumer requirements: An exploratory study of nineteen companies. In *21^{st} IEEE International Requirements Engineering Conference (RE'13)*, pages 105–114. IEEE, 2013.

[TT06] Toshihiko Tsumaki and Tetsuo Tamai. Framework for matching requirements elicitation techniques to project characteristics. *Software Process: Improvement and Practice*, 11(5):505–519, 2006.

[Tuu03] Tuure Tuunanen. A new perspective on requirements elicitation methods. *Journal of Information Technology Theory & Application*, 5(3):45–62, 2003.

[vDK99] Arie van Deursen and Tobias Kuipers. Identifying objects using cluster and concept analysis. In *21st ACM/IEEE International Conference on Software Engineering (ICSE'99)*, pages 246–255. ACM, 1999.

[vL09] Axel van Lamsweerde. *Requirements engineering: from system goals to UML models to software specifications*. Wiley, 2009.

[Vou08] Mladen A. Vouk. Cloud computing–issues, research and implementations. *Journal of Computing and Information Technology (CIT)*, 16(4):235–246, 2008.

[VS11] David Villegas and Seyed Masoud Sadjadi. Mapping non-functional requirements to cloud applications. In *23rd International Conference on Software Engineering and Knowledge Engineering (SEKE'11)*, pages 527–532, 2011.

[WB13] Karl Wiegers and Joy Beatty. *Software requirements*. Pearson Education, 2013.

[WH06] Roel J. Wieringa and Hans Heerkens. The methodological soundness of requirements engineering papers: a conceptual framework and two case studies. *Requirements Engineering (RE)*, 11(4):295–307, 2006.

[Wil09] Rudolf Wille. Restructuring lattice theory: An approach based on hierarchies of concepts. In *Formal Concept Analysis*, volume 5548 of *Lecture Notes in Computer Science (LNCS)*, pages 314–339. Springer, 2009.

[WMMR06] Roel Wieringa, Neil Maiden, Nancy Mead, and Colette Rolland. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requirements Engineering (RE)*, 11(1):102–107, 2006.

[WRH+12] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[YT03] Heng-Li Yang and Jih-Hsin Tang. A three-stage model of requirements elicitation for Web-based information systems. *Industrial Management & Data Systems*, 103(6):398–409, 2003.

[YWK+08] Da Yang, Di Wu, Supannika Koolmanojwong, A. Winsor Brown, and Barry W. Boehm. WikiWinWin: A wiki based system for collaborative requirements negotiation. In *41st Annual Hawaii International Conference on System Sciences (HICSS'08)*, pages 24–24. IEEE, 2008.

[ZB11] Shehnila Zardari and Rami Bahsoon. Cloud adoption: a goal-oriented requirements engineering approach. In *2nd International Workshop on Software Engineering for Cloud Computing*, pages 29–35. ACM, 2011.

[ZBE14] Shehnila Zardari, Rami Bahsoon, and Aniko Ekart. Cloud adoption: Prioritizing obstacles and obstacles resolution tactics using AHP. In *29th ACM Symposium on Applied Computing, Requirements Engineering Track*, pages 1013–1020. ACM, 2014.

[ZC05] Didar Zowghi and Chad Coulin. Requirements elicitation: A survey of techniques, approaches, and tools. In *Engineering and Managing Software Requirements*, pages 19–46. Springer, 2005.

[ZRB15] Ana Sofia Zalazar, Sebastian Rodriguez, and Luciana Ballejos. Handling dynamic requirements in cloud computing. In *Simposio Argentino de Ingeneria de Software (ASSE'15)*, pages 220–233, 2015.

Appendix A
Publications

This appendix presents the list of publications that represent the core of this dissertation.

A.1 Journal Article

[TKG15a] Irina Todoran Koitz and Martin Glinz. A Fuzzy Galois Lattices Approach to Requirements Elicitation for Cloud Services. In *IEEE Transactions on Services Computing (TSC)*, Published online, 2015. DOI: 10.1109/TSC.2015.2466538.

A.2 Conference Papers

[TSG13] Irina Todoran, Norbert Seyff, and Martin Glinz. How Cloud Providers Elicit Consumer Requirements: An Exploratory Study of Nineteen Companies. In *Proceedings of the 21st IEEE International Requirements Engineering Conference (RE'13)*. DOI: 10.1109/RE.2013.6636710.

[TG14] Irina Todoran and Martin Glinz. Quest for Requirements: Scrutinizing Advanced Search Queries for Cloud Services with Fuzzy Galois Lattices. In *Proceedings of the 10th IEEE World Congress on Services (SERVICES'14)*. DOI: 10.1109/SERVICES.2014.49.

[TKG15b] Irina Todoran Koitz and Martin Glinz. StakeCloud Tool: From Cloud Consumers' Search Queries to New Service Requirements. In *Proceedings of the 23rd IEEE International Requirements Engineering Conference (RE'15)*. DOI: 10.1109/RE.2015.7320441.

A.3 Other Publications (Not Included in the Dissertation)

[Tod12] Irina Todoran. StakeCloud: Stakeholder Requirements Communication and Resource Identification in the Cloud. In *Proceedings of the 20th IEEE International Requirements Engineering Conference (RE'12)*. Doctoral Symposium Paper. DOI: 10.1109/RE.2012.6345837.

[TG12] Irina Todoran and Martin Glinz. Towards Bridging the Communication Gap Between Consumers and Providers in the Cloud. In *Proceedings of the 10th Working IEEE/IFIP Conference on Software Architecture (WICSA/ECSA'12)*. Poster Paper. DOI: 10.1145/2361999.2362012.

[STC+15] Norbert Seyff, Irina Todoran, Kevin Caluser, Leif Singer, and Martin Glinz. Using Popular Social Network Sites to Support Requirements Elicitation, Prioritization and Negotiation. In *Journal of Internet Services and Applications (JISA)*, 2015. DOI: 10.1186/s13174-015-0021-9.

Curriculum Vitae

Name:	Irina Koitz (born Todoran)
Date of Birth:	May 25, 1986
Citizenship	Romania

Education

2011 - 2016	PhD in Computer Science - Requirements Engineering, University of Zurich, Switzerland
2011	Exchange Semester for Master Thesis, Munich University of Technology (TUM), Germany
2009 - 2011	Master in Computer Science - Service Design and Engineering, Aalto University (Helsinki University of Technology), Finland
2005 - 2009	Bachelor in Computer Engineering, Polytechnic University of Bucharest, Romania
2001 - 2005	High School, National College Ion Luca Caragiale, Ploiești, Romania

Professional Experience

2011 - 2016	Research Assistant - University of Zurich, Switzerland
2011	Researcher - Center for Digital Technology and Management (CDTM), Munich, Germany
2009 - 2011	Project Manager - Nokia Siemens Networks, Helsinki, Finland
2009	Intern, Microsoft, Bucharest, Romania